Learning Latent Variable and Predictive Models of Dynamical Systems

Sajid M. Siddiqi

CMU-RI-09-39

October 2009

Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Geoffrey J. Gordon, Chair
Andrew W. Moore
Jeff Schneider
Zoubin Ghahramani, University of Cambridge
David Wingate, Massachusetts Institute of Technology

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

This research was sponsored by NSF (ACI-0121671, 010543-1, 0000164), DARPA (F30602-01-2-0550, NBCHD030010), CDC (1R01PH00002801), USDA (4400161514, 4400147018, 4400150944), USAF (FA8650-05-C-7264), PA Department of Health (6003221), and a project with MobileFusion/TTC.

To my wife, Nada.

Acknowledgments

Despite the single author listed on the cover, this dissertation is not the product of one person alone. I would like to acknowledge many, many people who influenced me, my life and my work. They have all aided this research in different ways over the years and helped it come to a successful conclusion. Geoff Gordon, my advisor, has taught me a lot over the years; how to think methodically and analyze a problem, how to formulate problems mathematically, and how to choose interesting problems. From the outset, he has helped me develop the ideas that went into the thesis. Andrew Moore, my first advisor, got me started in machine learning and data mining and helped make this field fun and accessible to me, and his guidance and mentoring was crucial for work done early in my Ph.D. Both Geoff and Andrew are the very best kind of advisor I could have asked for: really smart, knowledgeable, caring and hands-on. They showed me how be a good researcher while staying relaxed, calm and happy. Though I wasn't always able to strike that balance, the example they set was essential for me to be able to make it through without burning out in the process.

All the members of the AUTON lab deserve much thanks, especially Artur Dubrawski and Jeff Schneider, as well as Mike Baysek for being the best system administrator anyone could ever work with. The SELECT lab, with which I became involved halfway through my Ph.D., was a very stimulating environment that helped enrich my research and learning. Thanks to Carlos Guestrin for the many insights and the keen intuition he added in lab meetings and personal discussions. I also had many fruitful discussions with Zoubin Ghahramani during his visits to Pittsburgh, who, with pen and paper, would present complex and exciting ideas in the most intuitive and accessible fashion. I would also like to thank Suzanne Lyons-Muth, Karen Widmaer, Sumitra Gopal and Jean Harpley for being great administrative coordinators and helping me with a smile whenever needed. For me, these people made the Robotics Institute the best place to be at Carnegie Mellon.

I would also like to acknowledge my friends, colleagues and collaborators here at Carnegie Mellon who helped make it such a great environment to work and learn in. A special thanks goes to Byron Boots, who is a collaborator on work presented in Chapters 5 and 6 in this thesis and from whom I have learned a lot about research and science. Abhishek Sharma, Sanjeev Koppal, Purnamrita Sarkar, Khalid El-Arini, Byron, Mike Stilman and David Choi have been my closest friends here over the years and were always there for me when I needed them. My memories of CMU would be incomplete without them. Thanks also to Ankur Datta, Obaidullah Khawaja, Julian Ramos, Mahesh Sabhnani, Miroslav Dudik, Ajit Singh, Stanislav Funiak, Brigham Anderson, Paul Komarek, Jeanie Komarek, Karen Chen, Anna Goldenberg, Jeremy Kubica, Gaurav Veda, Jonathan Huang, Sue Ann Hong and Joey Gonzalez who have been have been great friends, labmates and project partners throughout my time here. Much thanks to Stano for saving me from the rock I was stranded on during a SELECT lab whitewater rafting trip. I have learned a lot about life and research by interacting with all these people, as well as others, during my memorable time here at CMU.

My life in Pittsburgh would have been much more dreary without a larger community of friends which helped make Pittsburgh feel like home. Nader Shakir, Sarah Aboutalib, Khalid Harun, Faheem Hussain, Ziad Al-Bawab, Mudassir Alam, Mariam Mahmood, Rozana Hussin, Ranim Khalifa, Farah Khalifa, Basil Al-Sadhan, Osmaan Akhtar and others helped me have a more fulfilling social and spiritual life during my time in Pittsburgh. Those who were still in Pittsburgh in early 2008 provided a welcoming community of friends for my wife after she moved here, which was invaluable during the long days and weeks I was immersed in thesis work during the ensuing months. I am deeply grateful to them for having been such good friends to us.

Going further back, I would like to thank Gaurav Sukhatme at the University of Southern California for letting me work in his robotics lab, and for inducting me into the world of research in the wonderful environment of his research group at USC. Without his mentoring, encouragement and advice I would not have discovered the joys of scientific research in robotics and artificial intelligence, and I would not be here today. Srikanth Saripalli, Andrew Howard, Jim Montgomery, Kale Harbick, Boyoon Jung, David Naffin, Denis Wolf, Stefan Hrabar and Gabe Sibley were great colleagues during my time in Gaurav's lab. Ming-Deh Huang and Leonard Adleman were professors at USC who I had the privilege of taking classes with, who gave me the confidence to embark on my graduate studies.

My best friends at USC, Srikant Vaithilingam, Sanjeev Koppal (who I followed to CMU), Meng Lay and Faraz Husain were instrumental in making college a great experience, and we are all still close. Nathalie Collins, Madhura Sahasrabudhe, Rahul Joshi, Abrar Ahmed, Damion Desai, Omar Khan, and Jen Soon Ng were other close friends at CMU who I shared many good times with. Thank you all for helping make college life sane and happy.

Close friends from even further back during my childhood in Jeddah, and afterwards during my brief stays in Montreal and Delhi, also deserve a special mention for all they have given me over the years, especially Sujit Shankar, Ashraf Khan, Farhath Yar Khan, Shiraz Mahdi, Sarosh Mohiyuddin and Insha Naqvi. I also had many good memories and formative experiences with Syed Asim Ali, Imtiyaz R. Khan, Shadab Khan, Shemna Sanam, Rohini Mukherji, Khadija Ali Omer, Sikandar Siddiqui, Sharad Goel, Samad Hussain, Khizr Rathore, Tarique Siddiqui, Aarij Wasti, Shobhit Mehrotra and Mohit Gupta. Thank you for everything.

My family . . . Abbu, Ammi, Bhaijan, Bhayya, Apia and Bajia . . . nurtured and cared for me throughout my life, taught me everything important that I know, helped me study and do my homework on time, and footed my bills all the way. They were also the most supportive family throughout my Ph.D. despite the fact that it took me far away from them all the way to Pittsburgh, for the last 6 years. I am very grateful to them and love them all very much. I am especially thankful to Abbu and Ammi (who raised 5 children and made it look easy) for giving me immeasurable amounts of love and attention throughout my life, setting a formidably high bar for parenthood in the process. I'm also proud to be only the second Siddiqi in the family to earn a Ph.D., after Abbu. I'm thankful to God for giving me such wonderful and loving parents, brothers and sisters. Over the last few years, I have also had the good fortune of becoming an uncle to the sweetest nieces and nephews one could hope for. Adnan, Farhan, Sarah, Hana, Faraz, Faisal, Faiz, Ahmad and Sana: you are the greatest kids. I pray that you have the happiest lives.

My extended family, particularly Naana Jaan and Naani Amma, Phuppa and Phuppi, Chachcha, Chachchi, Ahmad Bhai and his family, and all my aunts and uncles have always been caring and supportive, and have always prayed for my success in whatever I do. I wish Naana Jaan and Chachchi were here today. I am forever indebted to my entire family for their love and support.

I am also fortunate to have such caring in-laws: Mummy, Pappa and Sana, and Hisham Bhai, Essam Bhai and both their families, have all been extremely loving, supportive and hospitable in these last few months. These months have been much more pleasant as a result, and I am grateful to them for including me wholeheartedly into their family.

Finally, Nada. I cannot overstate how much she has transformed my life for the better in the last few years, first as my friend, then as my best friend, and now as my wife. She has put her heart and soul into helping me complete this degree in one piece, and has sacrificed greatly in terms of her time, effort and tolerance for my constant busy-and-stressed-out status during the last few months. There is absolutely no doubt in my mind that I would have faltered near the end had she not stepped in and changed everything. Thank you. I love you very very much.

Contents

List of Figures

List of Tables

Abstract

A variety of learning problems in robotics, computer vision and other areas of artificial intelligence can be construed as problems of learning statistical models for dynamical systems from sequential observations. Good dynamical system models allow us to represent and predict observations in these systems, which in turn enables applications such as classification, planning, control, simulation, anomaly detection and forecasting. One class of dynamical system models assumes the existence of an underlying hidden random variable that evolves over time and emits the observations we see. Past observations are summarized into the belief distribution over this random variable, which represents the state of the system. This assumption leads to 'latent variable models' which are used heavily in practice. However, learning algorithms for these models still face a variety of issues such as model selection, local optima and instability. The representational ability of these models also differs significantly based on whether the underlying latent variable is assumed to be discrete as in Hidden Markov Models (HMMs), or real-valued as in Linear Dynamical Systems (LDSs). Another recently introduced class of models represents state as a set of predictions about future observations rather than as a latent variable summarizing the past. These 'predictive models', such as Predictive State Representations (PSRs), are provably more powerful than latent variable models and hold the promise of allowing more accurate, efficient learning algorithms since no hidden quantities are involved. However, this promise has not been realized.

In this thesis we propose novel learning algorithms that address the issues of model selection, local minima and instability in learning latent variable models. We show that certain 'predictive' latent variable model learning methods bridge the gap between latent variable and predictive models. We also propose a novel latent variable model, the Reduced-Rank HMM (RR-HMM), that combines desirable properties of discrete and real-valued latent-variable models. We show that reparameterizing the class of RR-HMMs yields a subset of PSRs, and propose an asymptotically unbiased predictive learning algorithm for RR-HMMs and PSRs along with finite-sample error bounds for the RR-HMM case. In terms of efficiency and accuracy, our methods outperform alternatives on dynamic texture videos, mobile robot visual sensing data, and other domains.

Chapter 1

Introduction

Modeling of dynamical systems is an important aspect of robotics, artificial intelligence and statistical machine learning. Such modeling is typically based on *observations* that arise from the dynamical system over time. A distinguishing characteristic of dynamical systems is that their observations exhibit *temporal correlations*, modeling of which is the main challenge of dynamical systems analysis. Accurate models of dynamical systems allow us to perform a variety of useful tasks, such as *prediction*, *simulation*, *recognition*, *classification*, *anomaly detection* and *control*. In this thesis we focus on *uncontrolled* dynamical systems with *multivariate real-valued observations*. This thesis contributes (1) novel learning algorithms for existing dynamical system models that overcome significant limitations of previous methods, (2) a deeper understanding of some important distinctions between different dynamical system models based on differences in their underlying assumptions and in their learning algorithms, and (3) a novel model that combines desirable properties of several existing models, along with inference and learning algorithms which have theoretical performance guarantees.

Two major approaches for modeling dynamical systems in machine learning are *Latent Variable Models (LVMs)* and *predictive models*, which have different benefits and drawbacks. An LVM for dynamical systems assumes its observations are noisy emissions from an underlying *latent variable* that evolves over time and represents the *state* of the system.

In other words, the latent state is a sufficient statistic for all *past* observations. LVMs probabilistically model both the latent variable's evolution and the relationship between latents and observables. Typical parameter learning algorithms for LVMs (such as the Expectation Maximization (EM) algorithm and related methods) are prone to *local optima* of their objective functions, and so cannot provide consistent parameter estimates with reasonable amounts of computation especially for large models, since multiple restarts are required to search the space of local optima. In contrast, predictive models (such as Predictive State Representations (PSRs)) and their learning algorithms define the state of a dynamical system as a set of *predictions* of expected values of statistics of the future, called *tests*. Since there are no latent or "hidden" quantities involved, learning algorithms for predictive models (which typically rely on matrix factorization rather than on EM) can yield *consistent* parameter estimates, though guaranteeing well-formed parameters with finite samples is often a challenge. Research from control theory as well as recent work in statistical learning theory (including parts of this thesis) have blurred the distinction between LVMs and predictive models by showing that LVMs can be learned in a globally optimal fashion with predictive algorithms, allowing us to interpret their latent variables as tests.

The two best-known examples of LVMs for continuous-observation dynamical systems are *Hidden Markov Models (HMMs)* (Chapter 2) and *Linear Dynamical Systems (LDSs)* (Chapter 3). Other LVMs for dynamical systems are often based on one or both of these two models. We describe important properties of HMMs and LDSs in more detail below.

HMMs assume a *discrete* latent variable that can take on finitely many values, each characterized by a unique probability distribution over observations. These assumptions allow HMMs to model a large variety of predictive distributions during forward simulation (including predictive distributions which are not log-concave), which is an advantage for modeling a variety of real-world dynamical systems. We will use the term *competitive inhibition* to denote the ability of a model (such as the HMM) to represent non-log-concave predictive distributions. A model that performs competitive inhibition can probability mass on distinct observations while disallowing mixtures of those observations. However, the discrete nature of the HMM's latent state makes it difficult to model smoothly evolving dynamical systems, which are also common in practice. Another difficulty with HMMs is the problem of *model selection*, or determining the correct number of states and the structure (e.g. sparsity) of the transition and observation functions.

On the other hand, LDSs assume a *continuous* latent variable that evolves linearly with Gaussian noise, and a Gaussian observation distribution whose mean is linear in the latent variable. These assumptions make LDSs adept at modeling smoothly evolving dynamical systems but unable to perform competitive inhibition. The inability to handle competitive inhibition stems from the fact that the predictive distribution is always log-concave; therefore any convex combination of likely observations will also be likely. Also unlike HMMs, matrix-factorization-based approaches to learning LDSs make it easy to perform model selection. Another distinction from HMMs is that conventional learning algorithms for the LDS do not guarantee *stable* parameters for modeling its dynamics. This can be either a benefit or a drawback, since the system to be modeled may be either unstable or stable. However, all of the systems that we consider modeling in this thesis are stable, so we consider it a drawback when a learning algorithm returns an unstable set of parameters.

In this thesis we advance the theory and practice of learning dynamical system models from data in several ways. We first address the tendency of HMM learning algorithms to get stuck in local optima, and the need to pre-define the number of states: we develop *Simultaneous Temporal and Contextual Splitting* (STACS), a novel EM-based algorithm for performing both model selection and parameter learning efficiently in Gaussian HMMs while avoiding local minima (Chapter 4). Results show improved learning performance on a wide variety of real-world domains. We next address a deficiency in conventional LDS learning algorithms: we propose a matrix-factorization-based learning algorithm for LDSs that uses *constrained optimization* to guarantee stable parameters and yields superior results in simulation and prediction of a variety of real-world dynamical systems (Chapter 5). Finally, we address the more ambitious goal of bridging the gap between models that can perform competitive inhibition and models that can represent smoothly evolving systems: we propose the *Reduced-Rank Hidden Markov Model (RR-HMM)*, a model that can do both the above (Chapter 6). We investigate its relationship to existing models, and propose a predictive learning algorithm along with theoretical performance

guarantees. We demonstrate results on a variety of high-dimensional real-world data sets, including vision sensory output from a mobile robot.

Chapter 2

Hidden Markov Models

Hidden Markov Models (HMMs) are LVMs where the underlying hidden variable can take on one of finitely many *discrete* values. Introduced in the late 1960s, HMMs have been used most extensively in speech recognition [4, 5] and bioinformatics [6] but also in diverse application areas such as computer vision and information extraction [7, 8]. For an excellent tutorial on HMMs, see Rabiner [4]. In this chapter we define HMMs and describe their standard inference and learning algorithms.

2.1 Definition

Let $h_t \in 1, \ldots, m$ denote the discrete hidden states of an HMM at time t, and $x_t \in 1, \ldots, n$ denote the observations. These can be either discrete or continuous—we will specify our assumptions explicitly for different instances. Let $T \in \mathbb{R}^{m \times m}$ be the state transition probability matrix with its $[i, j]^{th}$ entry T_{ij} having value $\Pr[h_{t+1} = i \mid h_t = j]$. O is the column-stochastic observation model such that $O(i, j) = \Pr[x_t = i \mid h_t = j]$. For discrete observations, O is a matrix of observation probabilities of size $n \times m$, and $O(i, j) = O_{ij}$ denotes the $[i, j]^{th}$ entry of O. For continuous observations, $O(x, j)$ denotes the probability of observation x under a Gaussian distribution specific to state j, i.e. $O(x, j) = \mathcal{N}(x \mid \mu_j, \Sigma_j)$. $\vec{\pi} \in \mathbb{R}^m$ is the initial state distribution with $\vec{\pi}_i = \Pr[h_1 = i]$. We use λ to denote the entire set of HMM parameters $\{T, O, \pi\}$. Figure 2.1 illustrates the graphical model corresponding to an HMM.

Let $\vec{h}_t \in \mathbb{R}^m$ denote the system's *belief state*, i.e. a distribution over hidden states at time t. If we use e_i to denote the i^{th} column of the identity matrix, then $\vec{h}_t$ is equivalent to the conditional expectation of e_{h_t}, with the conditioning variables clear from context. We use the term *path* to denote a sequence of hidden states $H = h_1, h_2, \ldots, h_\tau$ corresponding to a sequence of observations $X = x_1, x_2, \ldots, x_\tau$ from an HMM.

Computing the probability of a sequence of observations with an HMM is very simple. Note that $\Pr[x_1]$ can be expressed as a chain product of HMM parameters. Let $O_{x,\cdot}$ denote the row of O containing probabilities for observation x under each possible state. Now, define the parameters A_x as

$$A_x = T \ \operatorname{diag}(O_{x,\cdot}) \tag{2.1}$$

Then,

$$\Pr[x_1] = \sum_g \Pr[x_1 \mid h_1 = g] \Pr[h_1 = g] = \vec{1}_m^\mathsf{T} \ \operatorname{diag}(O_{x_1,\cdot})\vec{\pi}$$

Similarly, $\Pr[x_1 x_2 \ldots x_\tau]$ can be expressed as

$$\begin{aligned}
&\sum_{g_\tau,\ldots,g_1} \Pr[x_\tau \mid h_\tau = g_\tau] \Pr[h_\tau = g_\tau \mid h_{\tau-1} = g_{\tau-1}] \cdots \Pr[x_1 \mid h_1 = g_1] \Pr[h_1 = g_1] \\
&= \vec{1}_m^\mathsf{T} T \ \operatorname{diag}(O_{x_\tau,\cdot}) T \ \operatorname{diag}(O_{x_{\tau-1},\cdot}) \cdots T \ \operatorname{diag}(O_{x_1,\cdot})\vec{\pi} \\
&= \vec{1}_m^\mathsf{T} A_{x_\tau} A_{x_{\tau-1}} A_{x_{\tau-2}} \ldots A_{x_1}\vec{\pi}
\end{aligned} \tag{2.2}$$

We now describe standard algorithms for filtering (forward inference), smoothing (backward inference), path inference, learning and model selection in HMMs.

2.2 Filtering and Smoothing

Filtering is the process of maintaining a belief distribution while incrementally conditioning on observations over time in the forward direction, from past to present. Define the

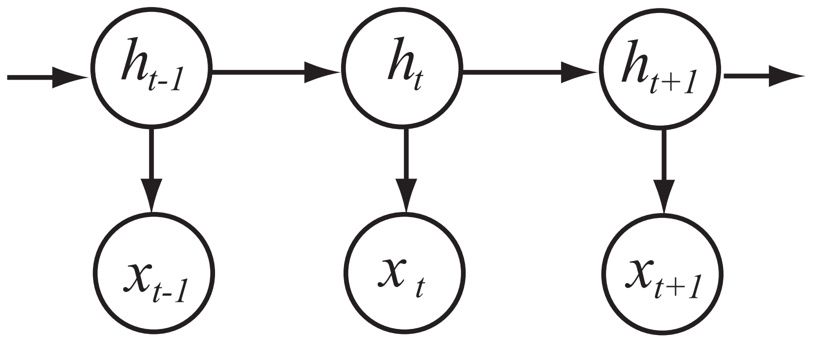


Figure 2.1: The graphical model representation of an HMM.

forward variable $\alpha(t,i)$ as

$$\alpha(t,i) = \Pr[x_1,\ldots,x_t,h_t = i \mid \lambda]$$

Then, the filtering belief distribution can be written using Bayes rule as the vector $\overline{\alpha}(t,\cdot) = [\overline{\alpha}(t,1)\ldots\overline{\alpha}(t,m)]^{\mathsf{T}}$ where the i^{th} element is:

$$\overline{\alpha}(t,i) \equiv \Pr[h_t = i \mid x_1,\ldots,x_t,\lambda] = \frac{\alpha(t,i)}{\sum_j \alpha(t,j)}$$

The values of $\alpha(t,i)$ for all t,i can be computed inductively according to the following equation [4]:

$$\alpha(t,i) = O(i,x_t)\sum_j \alpha(t-1,j)T_{ij}$$

The corresponding vector update equation is:

$$\alpha(t,\cdot) = \operatorname{diag}(O(:,x_t))T\alpha(t-1,\cdot)$$

Given the filtering belief probability $\overline{\alpha}(t,i)$ of being in state i at time t , the corresponding probability at time $t+1$ can be obtained directly in the following two steps:

$$\begin{aligned}
&\Pr[h_{t+1} = i \mid x_1, \ldots, x_t] \\
&= \sum_j \Pr[h_{t+1} = i \mid h_t = j] \Pr[h_t = j \mid x_1, \ldots, x_t] \quad \text{(prediction step)} \\
&= \sum_j T_{ij} \overline{\alpha}(t, j) \\
\overline{\alpha}(t+1, i) &= \Pr[h_{t+1} = i \mid x_1, \ldots, x_t, x_{t+1}] \\
&= \frac{\Pr[x_{t+1} \mid h_{t+1} = i] \Pr[h_{t+1} = i \mid x_1, \ldots, x_t]}{\sum_j \Pr[x_{t+1} \mid h_{t+1} = j] \Pr[h_{t+1} = j \mid x_1, \ldots, x_t]} \quad \text{(update step)} \\
&= \frac{O(i, x_{t+1}) \Pr[h_{t+1} = i \mid x_1, \ldots, x_t]}{\sum_j O(j, x_{t+1}) \Pr[h_{t+1} = j \mid x_1, \ldots, x_t]}
\end{aligned}$$

The forward variables also allow easy computation of the observation sequence probability:

$$\Pr[X \mid \lambda] = \sum_j \Pr[x_1, \ldots, x_\tau, h_\tau = j \mid \lambda] = \sum_j \alpha(\tau, j)$$

In contrast to filtering, *smoothing* in HMMs is the process of maintaining a belief distribution over the present based on observations in the future. Define the *backward variable* $\beta(t, i)$ as

$$\beta(t, i) = \Pr[h_t = i \mid x_{t+1}, \ldots, x_\tau]$$

The value of $\beta(t, i)$ can also be updated inductively as follows.

$$\beta(t, i) = \sum_j O(j, x_{t+1}) T_{ji} \beta(t+1, j)$$

The corresponding vector update equation is:

$$\beta(t, \cdot) = T^\mathsf{T} \ \mathrm{diag}(O(:, x_t)) \beta(t+1, \cdot)$$

The process of computing the forward and backward variables from an observation sequence using filtering and smoothing as described above is known as the *forward-backward algorithm*. Its running time is $\mathcal{O}(\tau m^2)$. Together, the forward and backward variables allow us to compute the *posterior stepwise beliefs* and *posterior stepwise transition probabilities* for an entire observation sequence, which are denoted by $\gamma(t,i)$ and $\xi(t,i,j)$ respectively:

$$\gamma(t,i) \equiv \Pr[h_t = i \mid x_1, \ldots, x_\tau] = \frac{\alpha(t,i)\beta(t,i)}{\Pr[X \mid \lambda]}$$
$$\xi(t,i,j) \equiv \Pr[h_t = i, h_{t+1} = j \mid x_1, \ldots, x_\tau] = \frac{\alpha(t,i)T_{ji}\beta(t+1,j)O(j,x_{t+1})}{\Pr[X \mid \lambda]}$$

As before, the denominators can be computed quickly by summing over the numerator. These variables will be useful when describing parameter learning algorithms for HMMs.

2.3 Path Inference

Path inference is the task of computing a path corresponding to a given observation sequence for a given HMM, such that the joint likelihood of path and observation sequence is maximized. Let λ denote the HMM and X denote the sequence of observations. Then, path inference computes H^* such that

$$H^* = \arg\max_H \Pr[X, H \mid \lambda]$$

For an observation sequence of length τ, the *Viterbi algorithm* [9] computes an optimal path in running time $\mathcal{O}(\tau m^2)$ using dynamic programming.

Define $\delta(t,i)$ as

$$\delta(t,i) = \max_{h_1,\cdots,h_{t-1}} \Pr[h_1 h_2 \cdots h_t = i, x_1 x_2 \cdots x_t \mid \lambda]$$

Though computing $\delta(t,i)$ for all t,i naively would have running time that is exponential in τ, it can be computed inductively in a more efficient fashion. The inductive formula for $\delta(t,j)$ used in the Viterbi algorithm is

$$\delta(t,j) = \left(\max_i \delta(t,i)T_{ij}\right) O(j, x_{t+1})$$

Since there is a maximization over m terms carried out for each state per timestep, and there are $m \times \tau$ $\delta(t,i)$ values to be calculated, the total running time of the Viterbi algorithm is $\mathcal{O}(\tau m^2)$.

2.4 Learning Hidden Markov Models

A large body of research exists on algorithms for learning HMM parameters from data. We focus on two of the most common techniques here, namely *Baum-Welch* and *Viterbi Training*. These are both iterative methods analogous to the popular k-means algorithm [10, 11] for clustering independent and identically distributed (IID) data, in the sense that they monotonically minimize a distortion function of the data with respect to a fixed number of "centers" (here, *states*) using successive iterations of computing distances to these centers and updating these centers to better positions. Since HMMs model sequential data, there is an additional dimension to these learning algorithms, namely the order in which datapoints tend to appear in the training sequence, which is modeled by the HMM's transition matrix.

The more recently developed *spectral learning algorithm* for HMMs [12, 13] relies on a Singular Value Decomposition (SVD) of a correlation matrix of past and future observations to derive an *observable representation* of an HMM. We describe this algorithm in detail in Chapter 6.

2.4.1 Expectation Maximization

Given one or several sequences of observations and a desired number of states m, we can fit an HMM to the data using an instance of EM [14] called *Baum-Welch* [15] which was discovered before the general EM algorithm. Baum-Welch alternates between steps of computing a set of expected sufficient statistics from the observed data (the *E-step*) and updating the parameters using estimates computed from these statistics (the *M-step*). The main advantage of Baum-Welch is that these closed-form iterations are guaranteed to monotonically converge to an optimum of the observed data log-likelihood $ll_X(\lambda) = \log \Pr[X \mid \lambda]$. The disadvantage is that it is only guaranteed to reach a *local* optimum, and there are no guarantees about reaching the global optimum. In practice, this issue is often addressed by running EM several times starting from different random parameter initializations. However, as the number of states increases, the algorithm is increasingly prone to local optima to an extent that is difficult to overcome by multiple restarts. The algorithm can be summarized as follows for a given training sequence $X = \langle x_1, x_2, ..., x_\tau \rangle$, for both discrete and continuous observations (assuming multinomial and Gaussian observation models respectively):

1. Initialize $\widehat{\lambda} = \langle \widehat{\pi}, \widehat{T}, \widehat{O} \rangle$ randomly to valid values (i.e. preserving non-negativity and stochasticity where needed).

2. Repeat while log-likelihood $ll(X \mid \widehat{\lambda})$ increases by more than some threshold ϵ:

   (a) E-step: Use forward-backward algorithm on $\widehat{\lambda}$ and X to compute $\alpha(t,i)$,$\beta(t,i)$ for all t, i and from these compute $\gamma(t,i)$ and $\xi(t,i,j)$ for all t, i, j.

   (b) M-step: Compute updated parameter estimates $\widetilde{\lambda} = \langle \widetilde{\pi}, \widetilde{T}, \widetilde{O} \rangle$ as follows:

$$\widetilde{\pi}_i = \gamma(1,i) \quad \forall i$$
$$\widetilde{T}(i,j) = \frac{\sum_t^{\tau-1} \xi(t,j,i)}{\sum_t^{\tau-1} \gamma(t,j)} \quad \forall i,j$$

   Multinomial observation model:

$$\widetilde{O}(i,x) = \frac{\sum_{t:x_t=x} \gamma(t,i)}{\sum_t^\tau \gamma(t,i)}$$

   Gaussian observation model:

$$\widetilde{\mu}_j = \frac{\sum_{t=1}^\tau \gamma(t,j) \cdot x_t}{\sum_{t=1}^\tau \gamma(t,j)} \quad \forall j$$
$$\widetilde{\Sigma}_j = \frac{\sum_{t=1}^\tau \gamma(t,j) \cdot (x_t - \mu_j)(x_t - \mu_j)^\mathsf{T}}{\sum_{t=1}^\tau \gamma(t,j)} \quad \forall j$$

   (c) $\widehat{\lambda} \leftarrow \widetilde{\lambda}$

3. Return final parameter estimates $\widehat{\lambda}$

2.4.2 Viterbi Training

Viterbi Training is the hard-updates analogue of Baum-Welch, in the sense that the E-step approximates the posterior stepwise belief and transition probability distributions γ and ξ with delta functions at a particular state and transition at every timestep. The particular state and transition chosen at each timestep are the state and transition in the *Viterbi path* at that time. The M-step therefore sets transition and observation probabilities based on counts computed from the Viterbi path. To update the prior π, if training is being performed using several observation sequences the prior is based on the distribution of h_1^* over these sequences. For a single training sequence, it is best to set the prior to be uniform rather than setting it to be a delta function at h_1^*, though intermediate choices are also possible (e.g. *Laplace Smoothing*, which allows biased priors while ensuring no probability is set to zero). The steps of Viterbi Training can be summarized as follows:

1. Initialize $\widehat{\lambda} = \langle \widehat{\pi}, \widehat{T}, \widehat{O} \rangle$ randomly to valid values (i.e. preserving non-negativity and stochasticity where needed).

2. Repeat while the Viterbi path keeps changing:

   (a) <u>E-step</u>: Compute the Viterbi path $H^* = \langle h_1^*, h_2^*, \ldots, h_\tau^* \rangle$

   (b) <u>M-step</u>: Compute updated parameter estimates $\widetilde{\lambda} = \langle \widetilde{\pi}, \widetilde{T}, \widetilde{O} \rangle$ as follows:

$$\widetilde{\pi}_i = \frac{1}{\tau} \quad \forall i$$
$$\widetilde{T}(i,j) = \frac{\sum_{t:h_t^*=j \wedge h_{t+1}^*=i} 1}{\sum_{t:h_t^*=j} 1} \quad \forall i,j$$

Multinomial observation model:

$$\widetilde{O}(i,x) = \frac{\sum_{t:h_t^*=i \wedge x_t=x} 1}{\sum_{t:h_t^*=i} 1}$$

Gaussian observation model:

$$\widetilde{\mu}_j = \frac{\sum_{t:h_t^*=j} x_t}{\sum_{t:h_t^*=j} 1} \quad \forall j$$

$$\widetilde{\Sigma}_j = \frac{\sum_{t:h_t^*=j} (x_t - \mu_j)(x_t - \mu_j)^\mathsf{T}}{\sum_{t:h_t^*=j} 1} \quad \forall j$$

(c) $\widehat{\lambda} \leftarrow \widetilde{\lambda}$

3. Return final parameter estimates $\widehat{\lambda}$

The asymptotic running time of both Baum-Welch and Viterbi Training is $\mathcal{O}(\tau m^2)$ per iteration. However, Viterbi Training is faster by a constant factor. Viterbi Training converges to a local maximum of the complete data likelihood $\Pr[X, H \mid \lambda]$, which does not necessarily correspond to a local maximum of the observed data likelihood $\Pr[X \mid \lambda]$ as is usually desired. In practice, Viterbi Training is often used to initialize the slower Baum-Welch algorithm which does converge to a local maximum of $\Pr[X \mid \lambda]$.

2.5 Related Work

Recently, HMMs and their algorithms have been re-examined in light of their connections to Bayesian Networks, such as in [16]. Many variations on the basic HMM model have also been proposed, such as coupled HMMs [7] for modeling multiple interacting processes, Input-Output HMMs [17] which incorporate inputs into the model, hierarchical HMMs [18] for modeling hierarchically structured state spaces, and factorial HMMs [19] that model the state space in a distributed fashion. Another notable example of a specialized sub-class of HMMs tailored for a particular task is the constrained HMM [20] which was developed originally in the context of speech recognition. Nonparametric methods such as Hierarchical Dirichlet Processes (HDPs) [21] have been used to define sampling-based versions of HMMs with “infinitely” many states [21, 22] which integrate out the hidden state parameter. This class of models has since been improved upon in several ways (e.g. [23]to bring it closer to a practical model, though it remains challenging to tractably perform learning or inference in these models on large multivariate data.

Chapter 3

Linear Dynamical Systems

In the case where the state of an LVM is *multivariate real-valued* and the noise terms are Gaussian, the resulting model is called a *linear dynamical system* (LDS), also known as a Kalman Filter [24] or a state-space model [25]. LDSs are an important tool for modeling time series in engineering, controls and economics as well as the physical and social sciences. In this section we define LDSs and describe their inference and learning algorithms as well as review the property of *stability* as it relates to the LDS transition model, which will be relevant later in Chapter 6. More details on LDSs and algorithms for inference and learning in LDSs can be found in several standard references [26, 27, 28, 29].

3.1 Definition

Linear dynamical systems can be described by the following two equations:

$$x_{t+1} = Ax_t + w_t \qquad w_t \sim \mathcal{N}(0, Q) \tag{3.1a}$$

$$y_t = Cx_t + v_t \qquad v_t \sim \mathcal{N}(0, R) \tag{3.1b}$$

Time is indexed by the discrete index t. Here x_t denotes the *hidden* states in $\mathbb{R}^n$, y_t the observations in $\mathbb{R}^m$, and w_t and v_t are Gaussian noise variables. In this thesis, we will assume w_t and v_t are zero-mean, though this may not hold in general. Assume the initial state $x(0) = x_0$. The parameters of the system are the dynamics matrix $A \in \mathbb{R}^{n \times n}$, the observation model $C \in \mathbb{R}^{m \times n}$, and the noise covariance matrices Q and R denoted by the following equation:

$$\mathbb{E}\left\{ \begin{bmatrix} w_t \\ v_t \end{bmatrix} \begin{bmatrix} w_s^\mathsf{T} & v_s^\mathsf{T} \end{bmatrix} \right\} = \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \delta_{ts} \tag{3.2}$$

In this thesis we are concerned with *uncontrolled* linear dynamical systems, though, as in previous work, control inputs can easily be incorporated into the model. Also note that in *continuous-time* dynamical systems, which we also exclude from consideration, the derivatives are specified as functions of the current state. They can be approximately converted to discrete-time systems via discretization.

3.2 Inference

In this section we describe the forwards and backwards inference algorithms for LDSs. More details can be found in several sources [26, 27, 29].

The distribution over state at time t, $\Pr[X_t \mid y_{1:T}]$, can be exactly computed in two parts: a forward recursion which is dependent on the initial state x_0 and the observations $y_{1:t}$ known as the *Kalman filter*, and a backward recursion which uses the observations from y_T to y_{t+1} known as the *Rauch-Tung-Striebel* (RTS) equations. The combined forward and backward recursions are together called the *Kalman smoother*. Finally, it is worth noting that the standard LDS filtering and smoothing inference algorithms [24, 30] are instantiations of the junction tree algorithm for Bayesian Networks on a dynamic Bayesian network analogous to the one in Figure 2.1 (see, for example, [31]).

3.2.1 The Forward Pass (Kalman Filter)

Let the mean and covariance of the belief state estimate $\Pr[X_t \mid y_{1:t}]$ at time t be denoted by $\hat{x}_t$ and $\hat{P}_t$ respectively. The estimates $\hat{x}_t$ and $\hat{P}_t$ can be predicted from the previous time step, the exogenous input, and the previous observation. Let $\hat{x}_{t_1|t_2}$ denote an estimate of variable x at time t_1 given data $y_1, \ldots, y_{t_2}$. We then have the following recursive equations:

$$x_{t|t-1} = Ax_{t-1|t-1} \tag{3.3a}$$
$$P_{t|t-1} = AP_{t-1|t-1}A^{\mathsf{T}} + Q \tag{3.3b}$$

Equation (3.3)(a) can be thought of as applying the dynamics matrix A to the mean to form an initial prediction of $\hat{x}_t$. Similarly, Equation (3.3)(b) can be interpreted as using the dynamics matrix A and error covariance Q to form an initial estimate of the belief covariance $\hat{P}_t$. The estimates are then adjusted:

$$x_{t|t} = x_{t|t-1} + K_t e_t \tag{3.3c}$$
$$P_{t|t} = P_{t|t-1} - K_t C P_{t|t-1} \tag{3.3d}$$

where the error in prediction at the previous time step (the innovation) e_t and the Kalman gain matrix K_t are computed as follows:

$$e_t = y_t - (C\hat{x}_{t|t-1}) \tag{3.3e}$$
$$K_t = P_{t|t-1}C^{\mathsf{T}}(C\hat{P}_{t|t-1}C^{\mathsf{T}} + R)^{-1} \tag{3.3f}$$

The weighted error in Equation (3.3)(c) corrects the predicted mean given an observation, and Equation (3.3)(d) reduces the variance of the belief by an amount proportional to the observation covariance. Taken together, Equations 3.3(a-f) define a specific form of the Kalman filter known as the *forward innovation model*.

3.2.2 The Backward Pass (RTS Equations)

The forward pass finds the mean and variance of the states x_t, conditioned on *past* observations. The backward pass corrects the results of the forward pass by evaluating the influence of *future* observations on these estimates. Once the forward recursion has completed and the final values of the mean and variance $x_{T|T}$ and $P_{T|T}$ have been calculated, the backward pass proceeds in reverse by evaluating the influence of future observations on the states in the past:

$$x_{t|T} = x_{t|t} + G_t(x_{t+1|T} - x_{t+1|t}) \tag{3.4a}$$

$$P_{t|T} = P_{t|t} + G_t(P_{t+1|T} - P_{t+1|t})G_t^{\mathsf{T}} \tag{3.4b}$$

where $x_{t+1|t}$ and $P_{t+1|t}$ are 1-step predictions

$$x_{t+1|t} = Ax_{t|t} \tag{3.4c}$$

$$P_{t+1|t} = AP_{t|t}A^{\mathsf{T}} + Q \tag{3.4d}$$

and the smoother gain matrix G is computed as:

$$G_t = P_{t|t}A^{\mathsf{T}}P_{t+1|t}^{-1} \tag{3.4e}$$

The cross variance $P_{t,t-1|T} = \mathrm{Cov}[X_{t-1}, X_t|y_{1:T}]$, a useful quantity for parameter estimation (section 3.3.1), may also be computed at this point:

$$P_{t-1,t|T} = G_{t-1}P_{t|T} \tag{3.4f}$$

3.3 Learning Linear Dynamical Systems

Learning a dynamical system from data (system identification) involves finding the parameters $\theta = \{A, C, Q, R\}$ and the distribution over hidden variables $\mathcal{Q} = P(X \mid Y, \theta)$ that maximizes the likelihood of the observed data. The maximum likelihood solution for these parameters can be found through iterative techniques such as expectation maximization (EM). An alternative approach is to use *subspace identification* methods to compute an asymptotically unbiased solution in closed form. In practice, a good approach is to use subspace identification to find a good initial solution and then to refine the solution with EM. The EM algorithm for system identification is presented in section 3.3.1 and subspace identification is presented in section 3.3.2.

3.3.1 Expectation Maximization

The EM algorithm is a widely applicable iterative procedure for parameter re-estimation. The objective of EM is to find parameters that maximize the likelihood of the observed data $P(Y \mid \theta)$ in the presence of latent variables x. EM maximizes the *log-likelihood*:

$$\mathcal{L}(\theta) = \log P(Y \mid \theta) = \log \int_X P(X, Y \mid \theta) dX \tag{3.5}$$

Using *any* distribution over the hidden variables $\mathcal{Q}$, a lower bound on the log-likelihood $\mathcal{F}(\mathcal{Q}, \theta) \leq \mathcal{L}(\theta)$ can be obtained by using Jensen's inequality (at equation (3.6b)). EM is derived by maximizing $\mathcal{F}(\mathcal{Q}, \theta)$ with respect to $\mathcal{Q}$, which results in $\mathcal{Q} = P(X \mid Y, \theta)$, the posterior over hidden variables given the data and current parameter settings:

$$\begin{aligned}
\mathcal{L}(\theta) &= \log P(Y \mid \theta) = \log \int_X P(X, Y \mid \theta) dX && \text{(3.6a)}\\
&= \log \int_X \mathcal{Q}(X) \frac{P(X, Y \mid \theta)}{\mathcal{Q}(X)} dX \geq \int_X \mathcal{Q}(X) \log \frac{P(X, Y \mid \theta)}{\mathcal{Q}(X)} dx && \text{(3.6b)}\\
&= \int_X \mathcal{Q}(X) \log P(X, Y \mid \theta) dX - \int_X \mathcal{Q}(X) \log \mathcal{Q}(X) dx && \text{(3.6c)}\\
&= \mathcal{F}(\mathcal{Q}, \theta) && \text{(3.6d)}
\end{aligned}$$

The EM algorithm alternates between maximizing the lower-bound on the likelihood $\mathcal{F}$ with respect to the parameters θ and the distribution $\mathcal{Q}$, holding the other quantity fixed. Starting from some initial parameters θ_0 we alternately apply:

$$\begin{aligned}
\text{Expectation-step (E-step): } \mathcal{Q}_{k+1} &\leftarrow \arg\max_{\mathcal{Q}} \mathcal{F}(\mathcal{Q}, \theta_k) && \text{(3.7a)}\\
\text{Maximization-step (M-step): } \theta_{k+1} &\leftarrow \arg\max_{\theta} \mathcal{F}(\mathcal{Q}_{k+1}, \theta) && \text{(3.7b)}
\end{aligned}$$

where k indexes an iteration, until convergence.

The E-Step The E-step is maximized when $\mathcal{Q}$ is exactly the conditional distribution of X, $\mathcal{Q}_{k+1}(X) = P(X \mid Y, \theta_k)$, at which point the bound becomes an equality: $\mathcal{F}(\mathcal{Q}_{k+1}, \theta_k) = \mathcal{L}(\theta)$. The maximum value of $\mathcal{Q}_{k+1}(X)$ can be found by solving the LDS inference (Kalman smoothing) problem: estimating the hidden state trajectory given the inputs, the outputs, and the parameter values. This algorithm is outlined in section 3.2.

The M-Step As noted in Equation (3.7)(b), the M-step is to find the maximum of $\mathcal{F}(\mathcal{Q}_{k+1}, \theta) = \int_X \mathcal{Q}_{k+1}(X) \log P(X, Y \mid \theta) dX - \int_X \mathcal{Q}_{k+1}(X) \log \mathcal{Q}_{k+1}(X) dx$ with respect to θ. The parameters of the system $\theta_{k+1} = \{\widehat{A}, \widehat{C}, \widehat{Q}, \widehat{R}\}$ are estimated by taking the corresponding partial derivative of the expected log-likelihood, setting to zero and solving, resulting in

$$\widehat{C} = \left(\sum_{t=1}^{T} y_t \mathbb{E}\{x_t^\top \mid y_{1:T}\} \right) \left(\sum_{t=1}^{T} \mathbb{E}\{x_t x_t^\top \mid y_{1:T}\} \right)^{-1} \tag{3.8a}$$

$$\widehat{R} = \frac{1}{T} \left(\sum_{t=1}^{T} y_t y_t^\top - \widehat{C} \sum_{t=1}^{T} \mathbb{E}\{x_t \mid y_{1:T}\} y_t^\top \right) \tag{3.8b}$$

$$\widehat{A} = \left(\sum_{t=2}^{T} \mathbb{E}\{x_t x_{t-1}^\top \mid y_{1:T}\} \right) \left(\sum_{t=2}^{T} \mathbb{E}\{x_{t-1} x_{t-1}^\top \mid y_{1:T}\} \right)^{-1} \tag{3.8c}$$

$$\widehat{Q} = \frac{1}{T-1} \left(\sum_{t=2}^{T} \mathbb{E}\{x_t x_t^\top \mid y_{1:T}\} - \widehat{A} \sum_{t=2}^{T} \mathbb{E}\{x_{t-1} x_t^\top \mid y_{1:T}\} \right) \tag{3.8d}$$

Also note that the state covariance matrix $\widehat{Q}$ (see Equation (3.8d)) is positive semi-definite by construction since it is the *Schur complement* [32] of

$$\sum_{t=1}^{T} \mathbb{E} \left\{ \begin{bmatrix} x_t x_t^\top & x_t x_{t-1}^\top \\ x_{t-1} x_t^\top & x_{t-1} x_{t-1}^\top \end{bmatrix} \mid y_{1:T} \right\} \geq 0 \tag{3.9}$$

3.3.2 Subspace Identification

Learning algorithms based on Expectation-Maximization (EM) iteratively optimize the *observed data likelihood* by (a) computing posterior estimates of first and second moments of the latent variable, and (b) computing the most likely parameters given these estimates of the latent variable. The Maximum-Likelihood Estimate (MLE) is *statistically efficient*, and EM-based methods can compute the MLE from finite data samples. However, EM-based methods are *computationally inefficient* because they only guarantee finding a *local* optimum of the observed data likelihood from a given starting point, and hence require multiple restarts from random parameter initializations to search for the MLE.

In contrast, *Subspace Identification* (Subspace ID) algorithms view the sequential data model learning task as being a *reduced-rank regression* from past to future observations,

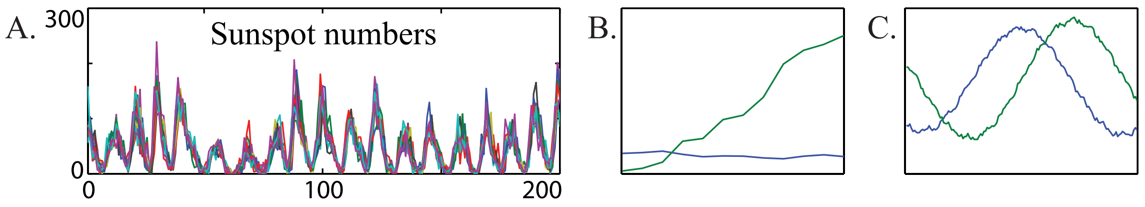


Figure 3.1: A. Sunspot data, sampled monthly for 200 years. Each curve is a month, the x-axis is over years. B. First two principal components of a 1-observation Hankel matrix. C. First two principal components of a 12-observation Hankel matrix, which better reflect temporal patterns in the data.

with the goal of minimizing the *predictive reconstruction error* in L_2 norm. The reason it is not an ordinary regression task is that the algorithm must compute a *subspace* of the observation space which allows prediction of future observations. This subspace is precisely the domain of the multivariate continuous latent state variable, and the parameters that map from this subspace to future latent states and observations are precisely the dynamics matrix and observation matrix of the LDS and products thereof. Note that, like other LVMs, since the LDS is an *unidentifiable* model, we only aim to discover the correct parameters up to a similarity transform. The benefits and drawbacks of Subspace ID are in some ways complementary to those of EM. Unlike multi-restart EM, Subspace ID is somewhat *statistically inefficient* since it does not achieve the MLE for finite data samples, but it is much more *computationally efficient* since it is not prone to local minima, and the *Singular Value Decomposition (SVD)* [32] attains the optimum parameter estimate efficiently in the limit. Thus at the granularity where SVD is a subroutine, Subspace ID is a non-iterative algorithm that admits a *closed-form solution*, though SVD internally is an iterative algorithm.

Subspace ID has been described clearly in the literature in several places such as Van Overschee (1996) [27], Katayama (2005) [29] and more recently in Boots (2009) [3]. We summarize the *uncontrolled* version of the algorithm here and refer the reader to the aforementioned references for details.

One useful degree of freedom that Subspace ID allows us is the ability to incorporate knowledge about the d*-step observability* of the linear system by specifying the length of observation sequences that are treated as features by the algorithm. This is accomplished by stacking observations in a *block Hankel matrix* [26] during regression, forcing the resulting parameter estimates to reconstruct entire *sequences* of observations based on multiple past observations. Define $Y_{0|i-1}$ as the following matrix of observations, where 0 and i are timesteps within the training data sequence:

$$Y_{0|i-1} = \begin{bmatrix} y_0 & y_1 & \cdots & y_{j-1} \\ y_1 & y_2 & \cdots & y_j \\ \vdots & \vdots & \ddots & \vdots \\ y_{i-1} & y_i & \cdots & y_{i+j-2} \end{bmatrix}_{mi \times j} \tag{3.10}$$

Y_p denotes a certain matrix of "past" observations, and Y_p^+ denotes its one-timestep *extension*. Also, Y_f, Y_f^- denote matrices of "future" inputs and observations and their one-step *contractions*:

$$Y_p \equiv Y_{0|i-1} \quad Y_p^+ \equiv Y_{0|i}$$
$$Y_f \equiv Y_{i|2i-1} \quad Y_f^- \equiv Y_{i+1|2i-1}$$

Matrices of the above form, with each block of rows equal to the previous block but shifted by a constant number of columns, are called *block Hankel* matrices [26]. We also define a matrix of Kalman filter latent state estimates at time i, *conditioned on past observations in* Y_p, as $\hat{X}_i$:

$$\hat{X}_i = [\hat{x}_i \; \hat{x}_{i+1} \; \ldots \; \hat{x}_{i+j}] \in \mathbb{R}^{n \times j} \tag{3.11}$$

Assuming the observations truly arise from an LDS, then the following relationship between expected future observations, latent states and LDS parameters must hold:

$$\mathbb{E}\{Y_f \mid \hat{X}_i\} = \begin{bmatrix} C\hat{x}_i & C\hat{x}_{i+1} & \cdots & C\hat{x}_{j-1} \\ C\hat{x}_{i+1} & C\hat{x}_{i+2} & \cdots & C\hat{x}_j \\ C\hat{x}_{i+2} & C\hat{x}_{i+3} & \cdots & C\hat{x}_{j+1} \\ \vdots & \vdots & \ddots & \vdots \\ C\hat{x}_{2i-1} & C\hat{x}_{2i} & \ldots & C\hat{x}_{2i+j-2} \end{bmatrix}_{mi \times j} \tag{3.12}$$

Γ_i is defined as the *extended observability matrix*:

$$\Gamma_i = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{i-1} \end{bmatrix}_{mi \times n} \tag{3.13}$$

Γ_i is related to its one-step contraction Γ_{i-1} by:

$$\Gamma_i = \begin{bmatrix} \Gamma_{i-1} \\ CA^{i-1} \end{bmatrix} \tag{3.14}$$

Using Γ_i from equation (3.13) in equation (3.12), $\mathbb{E}\{Y_f \mid \hat{X}_i\}$ can be written as:

$$\mathbb{E}\{Y_f \mid \hat{X}_i\} = \Gamma_i \hat{X}_i \tag{3.15}$$

Note that $\Gamma_i \hat{X}_i$ is a rank n linear function of state that lies in $\mathrm{span}\,\{Y_p\}$. The *linear projection* of Y_f onto Y_p may be used to find $\Gamma_i \hat{X}$ from Y_f and Y_p, where $\hat{X}$ denotes the Kalman filter states conditioned on observations in Y_p. These Kalman filter state estimates $\hat{X}_i$ can be computed exactly in closed form as a linear function of Y_p [27]. The projection is obtained by solving a set of linear equations

$$Y_f/Y_p = Y_f Y_p^\dagger Y_p = Y_f Y_p^\mathsf{T} (Y_p Y_p^\mathsf{T})^{-1} Y_p \tag{3.16}$$

where † denotes the Moore-Penrose pseudo-inverse [32]. Define $\mathcal{O}_i, \mathcal{O}_{i+1}$ as *projections of future observations onto past observations* in the following way:

$$\mathcal{O}_i = Y_f/Y_p = \Gamma_i \hat{X}_i \tag{3.17a}$$
$$\mathcal{O}_{i+1} = Y_f^-/Y_p^+ = \Gamma_{i-1} \hat{X}_{i+1} \tag{3.17b}$$

Subspace ID exploits relationships between several subspaces of interest to compute estimates of $\hat{X}_i$ and $\hat{X}_{i+1}$. The rank of $\mathcal{O}_i$ is the dimensionality of the state space, the row space of $\mathcal{O}_i$ is equal to the row space of $\hat{X}_i$ and the column space of $\mathcal{O}_i$ is equal to the column space of Γ_i [27]. Compute the SVD of $\mathcal{O}_i$:

$$\mathcal{O}_i = \mathcal{U} \Sigma \mathcal{V}^\mathsf{T} \tag{3.18}$$

By properties of SVD , we know the columns of $\mathcal{U}$ are an optimal basis for compressing and reconstructing sequences of i future observations.

As mentioned earlier, having multiple observations per column in Y_f is particularly helpful for learning models of systems that are not 1-step observable, e.g. when the underlying dynamical system is known to have periodicity and a single observation is not enough to tell us where we are in the period. For example, Figure 3.1(A) shows 200 years of sunspot numbers, with each month modeled as a separate variable. Sunspots are known to have two periods, the longer of which is 11 years. When subspace ID is performed using a 12-observation Hankel matrix Y_f and Y_p, the first two columns of $\mathcal{U}$, i.e. the first two principal components of $\mathcal{O}_i$, resemble the sine and cosine bases (Figure 3.1(C)), and the corresponding state variables therefore are the coefficients needed to combine these bases so as to reconstruct 12 years of the original sinusoid-like data, which captures their periodicity. This is in contrast to the bases obtained by SVD on a 1-observation Y_f and Y_p (Figure 3.1(B)), which reconstruct just the variation within a single year.

Though Y_p and Y_f have finitely many observations from past and future, this window size does not need to grow indefinitely for the algorithm to converge to the correct parameter estimates. For any given dynamical system, there's a minimum window length that will allow us to recover the true dynamics. The minimum window length is the shortest one in which we are guaranteed to have positive probability of observing something relevant to each dimension of latent state. Note that we don't need to guarantee a high probability of making such observations, or to guarantee that our observations are particularly informative about state, only that there is a positive probability of getting a positive amount of information. It is possible that the minimum window length is infinite, but then the dimensionality of the dynamical system would need to be infinite as well, hence it would a system that is difficult to recover with any method.

Subspace ID also allows a simple yet principled method of model selection: the order of the system can be determined by inspecting the *singular values* on the diagonal Σ, and all components whose singular values lie below a user-defined threshold can be dropped. One common way of choosing the dimensionality is to look for a "knee" in the graph of decreasing-order singular values, indicating the noisy data arises from a low-rank system. Then we estimate Γ_i, Γ_{i-1} and $\hat{X}_i$ as:

$$\Gamma_i = \mathcal{U}\Sigma^{1/2} \tag{3.19}$$

which, based on equation (3.14), allows us to estimate Γ_{i-1}:

$$\tilde{X}_i = \Gamma_i^{\dagger}\mathcal{O}_i \tag{3.20a}$$

$$\tilde{X}_{i+1} = \Gamma_{i-1}^{\dagger}\mathcal{O}_{i+1} \tag{3.20b}$$

Here we use $\tilde{X}_i$ to denote *estimates* of the *true* Kalman filter states $\hat{X}$ conditioned on observations in Y_p. An important result that allows Subspace ID to claim *consistency* is that, as the number of columns j in our Hankel matrices go to infinity, the state estimates converge to the true Kalman filter state estimates conditioned on observations in Y_p (up to a linear transform) [27]:

$$\tilde{X}_i \rightarrow \hat{X}_i$$
$$\tilde{X}_{i+1} \rightarrow \hat{X}_{i+1}$$

After computing the estimates $\tilde{X}_i$ and $\tilde{X}_{i+1}$ we can estimate the LDS parameters A and C by solving the following system of equations, which is straightforward:

$$\begin{bmatrix} \widehat{X}_{i+1} \\ Y_{i|i} \end{bmatrix} = \begin{bmatrix} A \\ C \end{bmatrix} \begin{bmatrix} \widehat{X}_i \end{bmatrix} + \begin{bmatrix} \rho_w \\ \rho_v \end{bmatrix} \tag{3.21a}$$

Here, ρ_w, ρ_v are the *residual errors* of the noisy LDS, which are assumed to have zero-mean Gaussian distributions. We can estimate the covariances $\widehat{Q}$ and $\widehat{R}$ from the estimated residuals:

$$\begin{bmatrix} \widehat{Q} & \widehat{S} \\ \widehat{S}^{\mathsf{T}} & \widehat{R} \end{bmatrix} = \mathbb{E}\left\{ \begin{bmatrix} \widehat{\rho}_w \\ \widehat{\rho}_v \end{bmatrix} \begin{bmatrix} \widehat{\rho}_w^{\mathsf{T}} \widehat{\rho}_v^{\mathsf{T}} \end{bmatrix} \right\} \tag{3.21b}$$

Since the state estimates converge to their true values, the parameter estimates $\theta = \{\widehat{A}, \widehat{C}, \widehat{Q}, \widehat{R}\}$ are asymptotically unbiased as the length of the training sequences goes to infinity [27].

3.4 Stability

We describe stability of dynamical systems based on material from [33]. Stability is a property of dynamical systems defined in terms of *equilibrium points*. If all solutions of a dynamical system that start out near an equilibrium state x_e stay near or converge to x_e, then the state x_e is stable or asymptotically stable respectively. A linear system $x_{t+1} = Ax_t$ (with zero noise) is *internally stable* if the matrix A is stable in the sense of Lyapunov (see below). Internal stability is sufficient, though not necessary, for the stability of a dynamical system. The standard algorithms for learning linear Gaussian systems described in Section 3.3 *do not enforce stability*; when learning from finite data samples, the maximum likelihood or subspace ID solution may be unstable even if the true system is stable due to the sampling constraints, modeling errors, and measurement noise. A square matrix A is said to be *asymptotically stable in the sense of Lyapunov* if and only if for a dynamics matrix A and any given positive semi-definite symmetric matrix Q there exists a positive-definite symmetric matrix P that satisfies the following *Lyapunov criterion*:

$$P - APA^{\mathsf{T}} = Q \tag{3.22}$$

There is a direct connection to this criterion and the Kalman filter update in Equation (3.3)(b). For a linear dynamical system, A is the dynamics matrix, P is the current belief covariance, and Q is the positive semi-definite state error covariance matrix (Equation (3.8d)). Thus, the Lyapunov criterion can be interpreted as holding if there exists a belief distribution where the predicted belief over state is equivalent to the previous belief over state. It is interesting to note that the Lyapunov criterion holds *if and only if* the spectral radius $\rho(A) \leq 1$. Recall that a matrix M is positive definite (semi-definite) *iff* $z^{\mathsf{T}}Mz > 0$ (≥ 0) for all non-zero vectors z. Let λ be an left eigenvalue of A and ν be a corresponding eigenvector, giving us $\nu^{\mathsf{T}}A = \lambda\nu^{\mathsf{T}}$, then

$$\nu^{\mathsf{T}}Q\nu = \nu^{\mathsf{T}}(P - A^{\mathsf{T}}PA)\nu = \nu^{\mathsf{T}}P\nu - \nu^{\mathsf{T}}\lambda P\lambda\nu = \nu^{\mathsf{T}}P\nu(1 - |\lambda|^2) \geq 0 \tag{3.23}$$

since $\nu^{\mathsf{T}}P\nu \geq 0$, it follows that $|\lambda| \leq 1$ and thus $\rho(A) \leq 1$. When $\rho(A) < 1$, the system is asymptotically stable. To see this, suppose $D\Lambda D^{-1}$ is the eigen-decomposition of A, where Λ has the eigenvalues of A along the diagonal and D contains the eigenvectors. Then,

$$\lim_{k\to\infty} A^k = \lim_{k\to\infty} D\Lambda^k D^{-1} = D\left(\lim_{k\to\infty} \Lambda^k\right) D^{-1} = 0 \tag{3.24}$$

since it is clear that $\lim_{k\to\infty} \Lambda^k = 0$. If $\rho(A) = 1$, then A is stable but not asymptotically stable, and the state x_t oscillates around x_e indefinitely. However, this is true only under the assumption of zero noise. In the case of an LDS with Gaussian noise, a dynamics matrix with unit spectral radius would cause the state estimate to move steadily away from x_e. Hence such an LDS is asymptotically stable only when $\rho(A)$ is strictly less than one, and the matrix Q in equation (3.22) above is required to be positive definite. If $\rho(A) = 1$, the LDS is said to be *marginally stable*.

3.5 Related Work

The EM algorithm for LDS was originally presented in [34]. Auto-Regressive (AR), Moving Average (MA) and Auto-Regressive Moving Average (ARMA) models are simpler time series modeling methods that are provably subsumed by LDSs [25]. Nonlinear dynamical systems and their learning algorithms have also been studied, such as the *extended Kalman filter* [35, 36] which linearizes the nonlinear system around the state estimate at every step, allowing the approximate state distribution to remain Gaussian. An EM algorithm for learning nonlinear dynamical systems has also been proposed[16], which uses Extended Kalman Smoothing to compute the non-Gaussian conditional hidden state distribution over the nonlinear dynamical system, and Radial Basis Functions (RBFs) to represent the nonlinearities.

Chapter 4

Fast State Discovery and Learning in Hidden Markov Models

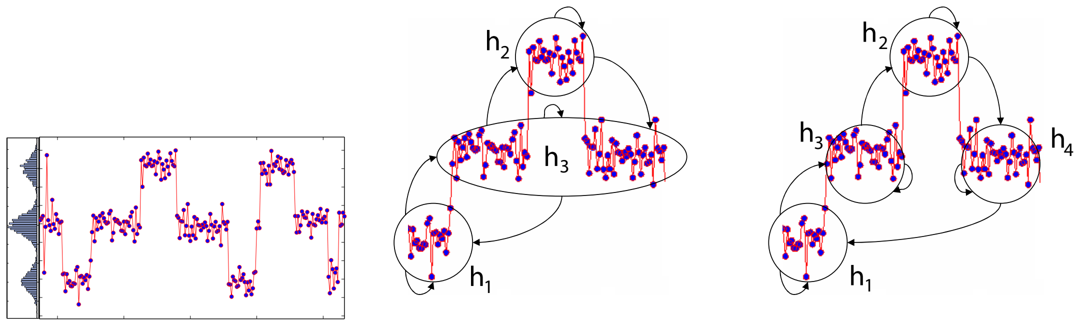
We first look at an algorithm for learning discrete-state LVMs with continuous observations, specifically Gaussian HMMs. Typical algorithms for learning HMMs rely on knowing the correct value for the number of states beforehand, and then optimizing the observed data likelihood for a fixed number of states, until a local optimum is reached. In contrast, STACS (Simultaneous Temporal and Contextual Splitting) reformulates the search space by incrementally increasing the number of states (by splitting an existing state) during parameter optimization in a way that maximally improves observed data likelihood. The algorithm terminates when the improvement in explanation of the data by further HMM growth no longer justifies the increase in model complexity, according to a standard model selection scoring criterion. Both the splitting and scoring processes are carried out efficiently by selective application of Viterbi approximations. This process makes parameter learning more efficient and also helps avoid the local minima which greatly hinder fixed-topology HMM learning algorithms, particularly for large state spaces.

4.1 Introduction

There has been extensive work on learning the parameters of a fixed-topology HMM. Several algorithms for finding a good number of states and corresponding topology have been investigated as well (e.g. [37, 38, 39]), but none of these are used in practice because of one or more of: *inaccuracy*, *high computational complexity*, or *being hard to implement*. Normally, the topology of an HMM is chosen *a priori* and a hill-climbing method is used to determine parameter settings. For model selection, several HMMs are typically trained with different numbers of states and the best of these is chosen. There are two problems with this approach: firstly, training an HMM from scratch for each feasible topology may be computationally expensive. Secondly, since parameter learning is prone to local minima, we may inadvertently end up comparing a 'good' m_1-state HMM to a 'bad' m_2-state HMM. The standard solution to this is to train several HMMs for each topology with different parameter initializations in order to overcome local minima, which however compounds the computational cost and is ineffectual for large state spaces. For example, Figure 4.1(A) shows a simple data sequence where many training algorithms can get caught in local minima. To illustrate the concepts we discuss, we shall use this example throughout this chapter.

Because of these issues, many researchers have previously investigated top-down state-splitting methods as an appealing choice for topology learning in HMMs with continuous observation densities. This chapter describes *Simultaneous Temporal and Contextual Splitting* (STACS), a recently proposed [40] top-down model selection and learning algorithm that constructs an HMM by alternating between parameter learning and model selection while incrementally increasing the number of states. Candidate models are generated by splitting existing states and optimizing relevant parameters, and are then evaluated for possible selection. Unlike previous methods, however, the splitting is carried out in a way that accounts for both *contextual* (observation density) and *temporal* (transition model) structure in the underlying data in a more general manner than the state-splitting methods mentioned above, which fail on the simple example in Figure 4.1(A). In this research, we closely examine these competing methods and illustrate the key differences between them and STACS, followed by an extensive empirical comparison. Since hard-updates training is a widely used alternative to soft-updates methods in HMMs because of its efficiency, we also examine a hard-updates version of our algorithm, *Viterbi STACS* (V-STACS), and explore its pros and cons for model selection and learning with respect to the soft-updates method. We also evaluate STACS as an alternative learning algorithm for models of *pre-determined* size. To determine the stopping point for state-splitting, we use the *Bayesian Information Criterion* [41], or BIC score. We discuss the benefits and drawbacks of this in Section 4.3.3. We compare our approach to previous work on synthetic data as well as several real-world data sets from the literature, revealing significant improvements in efficiency and test-set likelihoods. We compare to previous algorithms on a sign-language recognition task, with positive results. We also describe an application of STACS to learning models for detecting diverse events in real-world unstructured audio data.

Throughout this chapter, assume the HMM notation introduced in Chapter 2.



(A) (B) (C)

Figure 4.1: A. A time series from a 4-state HMM. Observations from the two states *down-middle-up* and *up-middle-down* overlap and are indistinguishable without temporal information. B. The HMM topology learned by ML-SSS and Li-Biswas on the data in A. C. The correct HMM topology, successfully learned by the STACS algorithm.

4.2 Related Work

There has been extensive work on HMM model selection. However, most of this work is either tailored to a specific application or is not scalable to learning topologies with more than a few states. The most successful approaches are greedy algorithms that are either bottom-up (i.e., starting with an overly large number of states) or top-down (i.e., starting with a small or single-state model). We will briefly discuss the bottom-up approaches and argue that they are unsuitable for practical large-scale HMM model selection as compared to top-down approaches. After this, we will discuss the major top-down approaches from the literature in more detail.

The primary drawbacks of bottom-up techniques are (a) having to choose and evaluate merges from among N^2 candidate pairs, (b) having to know the maximum number of states beforehand and (c) difficulty in generalizing to real-valued observations. Bottom-up topology learning techniques start off with a superfluously large HMM and prune parameters and/or states incrementally to shrink the model to an appropriate size. Stolke and Omohundro [42] demonstrate a Bayesian technique for learning HMMs by successively merging pairs of states for discrete-observation HMMs, followed by Baum-Welch to optimize parameter settings. Their model-merging technique starts off with one state for each unique discrete-valued observation, uses a heuristic to pick the top few merge candidates to evaluate. Another bottom-up approach is the Entropic Training technique of Brand [39]. This technique uses an entropy-based prior that favors simpler models and relies on an iteratively computed MAP estimator to successively trim model parameters. Though the algorithm applies to real-valued observations and does not require the computation of merges, it is complex, and the problem of having to start with a good upper bound on N still holds true.

We therefore favor top-down methods for HMM model selection, especially when the number of states may be large. We define some terminology first: *split design* refers to the process of splitting an HMM state, optimizing parameters and creating an HMM for possible selection. HMMs created by designing different splits are called *candidates*. The two major alternative top-down approaches can be summarized as follows:

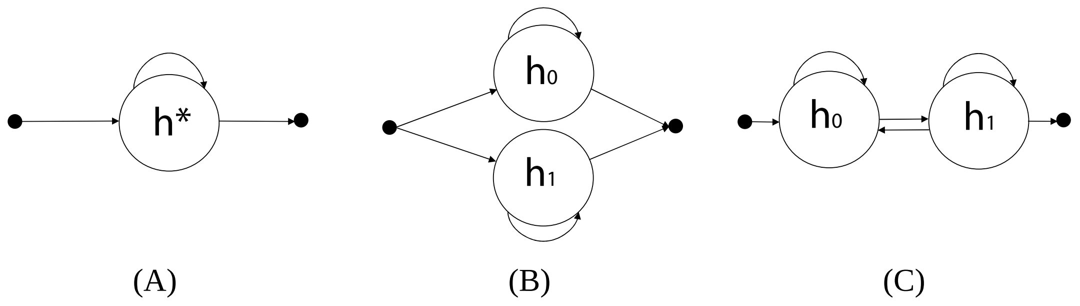


Figure 4.2: An illustration of the overly restrictive splits in ML-SSS. A. Original un-split state h^*. B. A *contextual split* of h^* in the ML-SSS algorithm. States h_0 and h_1 must have the *same* transition structures and different observation models. C. A *temporal split*. State h_0 has the incoming transition model of h^* and h_1 has its outgoing ones.

Li-Biswas: This algorithm [37] examines two model selection candidates at every step: one obtained by splitting the state with largest observation density variance, and the other by merging the two states whose means are closest in Euclidean space. These candidates are then optimized with EM on the entire HMM. The candidate with better likelihood is chosen at each step, terminating when the candidates are worse than the original model. The primary drawback with this heuristic is that it ignores dynamic structure while deciding which states to split: a single low-variance state might actually be masking two Markov states with overlapping densities, making it a better split candidate. Training two candidates with full EM is also inefficient especially when they may not be the best candidates, as our empirical evaluations will show.

ML-SSS: *Maximum-Likelihood Successive-State-Splitting* [38] is designed to learn HMMs that model variations in phones for continuous speech recognition systems. ML-SSS incrementally builds an HMM by splitting one state at a time, considering all m possible splits as candidates in each timestep. However, instead of full EM for each candidate, the split on state h^* into h_0 and h_1 (figure 4.2) is trained by a constrained iterative optimization of the expected likelihood gain from performing the split, while holding all $\gamma_t(h)$ and $\xi_t(h, h')$ constant for $h, h' \neq h^*$. The iterations are performed over all timesteps

t with non-zero[1] posterior occupancy probability for h^* . Each state is considered for two kinds of splits, a *contextual split* (Figure 4.2(B)) that optimizes only observation densities, and a *temporal split* (figure 4.2(C)) that also optimizes self-transition and inter-split-state transition probabilities.

Though more efficient than brute-force and Li-Biswas, the fact that ML-SSS does not model transitions to and from other states while splitting makes it fail to detect underlying Markov states with overlapping densities. For example, ML-SSS with BIC converges on the HMM in figure 4.1(B) while the true underlying HMM, successfully found by STACS, is shown in figure 4.1(C). Also, having to consider all data points with non-zero posterior probability $\gamma_t(h^*)$ on every split is expensive in dense data with overlapping densities.

4.3 Simultaneous Temporal and Contextual Splits

STACS is based on the insight that EM for sequential data is more robust to local minima for small state spaces (e.g. two states) but less so for large state spaces. STACS uses this insight to use a constrained two-state EM algorithm to break out of local minima and increase state space size. STACS algorithms perform parameter learning for continuous-density HMMs with Gaussian observation models, while simultaneously optimizing state space size and transition topology in a data-driven fashion. Unlike [38], our method accounts for both temporal and contextual variations in the training observations while splitting states (i.e. increasing the state space size). Unlike [37], our method evaluates every existing state as a possible candidate for splitting. Since naively evaluating all possible ways to increase state space size would be very expensive computationally, STACS algorithms make selective use of *Viterbi approximations* [42, 43] for efficient approximation of the data log-likelihood, as well as some other assumptions detailed below. These approximations keep the complexity of each iteration of STACS and V-STACS to be $\mathcal{O}(\tau m^2)$, with V-STACS being faster by a constant factor that is quite noticeable in practice. Here τ is the length of the training sequence and m is the number of states currently in the HMM.

[1]In practice, non-zero corresponds to being above a specified cutoff value

Experiments show that STACS outperforms alternative topology learning methods [38, 37] in terms of running time, test-set likelihood and BIC score on a variety of real-world data, and outperforms regular EM even on learning models of predetermined size. This top-down approach proved to be highly effective at avoiding local minima as well, allowing STACS to discover the true underlying HMM in difficult synthetic data where EM failed to find the correct answer even with 50 restarts. This highlights the problem with cross-validation for determining the number of states, which we alluded to in Section 4.1: due to the local minima problems rife in EM-based HMM learning, there is no way to ensure during cross-validation that the best possible HMMs of different sizes are being compared. STACS avoids this problem by guaranteeing a consistent increase in data likelihood as it searches the space of HMMs of varying state space sizes, based on monotonicity results regarding variants of the EM algorithm [44].

We describe the overall algorithm as well as details of STACS and V-STACS below. First, some notation: when considering a split of state h, HMM parameters related to state h (denoted by λ_h), including incoming and outgoing transition probabilities, are replaced by parameters for two offspring states h_1 and h_2 (denoted by λ_{h_1,h_2}). The time indices assigned to state h, denoted by $\tau(h)$, are now assumed to have unknown hidden state values, but only in the restricted state space $\{h_1, h_2\}$. Therefore when searching for a locally optimal candidate, only the parameters λ_{h_1,h_2} change, and the only observations that will affect them are those at timesteps $\tau(h)$, i.e., $X_{\tau(h)}$. Let $\lambda_{\setminus h}$ denote parameters not related to state h.

4.3.1 The Algorithm

STACS and V-STACS both have the following overall procedure. For each step that is not constant-time, we list the asymptotic complexities as $[\cdot]$ or as $[\cdot, \cdot]$ respectively if they differ. Details on candidate generation and candidate selection are given in subsequent sections.

1. Initialization: Initialize λ to a single-state HMM ($m = 1$) using X. $[\mathcal{O}(\tau)]$

2. Learning: Use Baum-Welch or Viterbi Training until convergence of $P[X \mid \lambda]$. $[\mathcal{O}(\tau m^2)]$

3. Candidate Generation: Split each state, to generate m candidate HMMs each with $m+1$ states. $[\mathcal{O}(\tau m^2), \mathcal{O}(\tau m)]$

4. Candidate Selection: Score the original HMM and each candidate, and pick the highest scoring one as λ'. $[\mathcal{O}(\tau m^2), \mathcal{O}(\tau m)]$

5. Repeat or Terminate: If a split candidate was picked, $\lambda \leftarrow \lambda', m \leftarrow m+1$ and go to step 2. Else if original HMM was picked, terminate and return λ'.

4.3.2 Generating Candidates

To generate a candidate based on state h resulting in new states h_1 and h_2, we devised two novel variants of EM to efficiently compute optimal means, variances and transition parameters of the resulting 2 states. We first perform Viterbi to find the optimal state sequence H^* by maximizing $P(X, H \mid \lambda)$. We then constrain the parameters for all other states ($\lambda_{\setminus h}$). We assume all timesteps belonging to other states in Q^* are associated with those states exclusively, which is equivalent to approximating the posterior belief at each timestep by a delta function at the Viterbi-optimal state. Then, we perform ***Split-State Viterbi Training*** (for V-STACS) or ***Split-State Baum-Welch*** (for STACS) to optimize λ_{h_1,h_2} on the timesteps associated with state h i.e. $X_{\tau(h)}$. This effectively optimizes a ***partially observed likelihood*** $P(X, H^*_{\setminus\tau(h)} \mid \lambda)$.

Candidate generation is highly efficient: Split-State Viterbi Training is $\mathcal{O}(|\tau(h)|)$, Split-State Baum-Welch is $\mathcal{O}(m|\tau(h)|)$. Since $|\tau(h)|$ is equal to τ/m on average, and there are m candidates to be generated, the total cost is $\mathcal{O}(m\tau)$ and $\mathcal{O}(\tau m^2)$ respectively.

Split-State Viterbi Training

Split-State Viterbi Training is the candidate generation algorithm for V-STACS. The algorithm learns locally optimal values for the parameters λ_{h_1,h_2} by alternating the following two steps for each iteration i until convergence:

<u>**E-step**</u> : $H^i_{\tau(h)} \leftarrow \arg\max_H P(X_{\tau(h)}, H^*_{\backslash\tau(h)}, H \mid \lambda_{\backslash h}, \lambda^i_{h_1,h_2})$

<u>**M-step**</u> : $\lambda^{i+1}_{h_1,h_2} \leftarrow \arg\max_\lambda P(X_{\tau(h)}, H^*_{\backslash\tau(h)}, H^i_{\tau(h)} \mid \lambda_{\backslash h}, \lambda)$

Here, $H^*_{\backslash\tau(h)}$ denotes the base model Viterbi path excluding timesteps belonging to state h. The first step above computes an optimal path through the split-state space. The second step updates the relevant HMM parameters with their MLE estimates for a fully observed path, which are simply ratios of counts for transition parameters, and averages of subsets of $X_{\tau(h)}$ for the observation parameters. Convergence occurs when the state assignments on $\tau(h)$ stop changing. The E-step of Split-State Viterbi Training is carried out using a novel adaptation of Viterbi (called *Split-State Viterbi*) to the task of finding an optimal path over a binary state space through a subset of the data while constraining the rest to specific states. Given below is pseudocode for the Split-State Viterbi algorithm.

<u>***Split-State Viterbi***</u>$(\lambda, H^*, \tau^h, X_{\tau^h})$

Initialization: Increase the number of states in the HMM by 1. Initialize new parameters for states h_1 and h_2 that replace state h. For all other states h' and for $i, j = 1, 2$:

```
  $\pi_{h_i} \leftarrow \frac{1}{2}\pi_h$
  $T_{h'h_i} \leftarrow \frac{1}{2}T_{h'h}$
  $T_{h_ih'} \leftarrow T_{hh'}$
  $T_{h_ih_j} \leftarrow \frac{1}{2}T_{hh}$
  $O_{h_i} \leftarrow$ initialize to MLE Gaussian using $X_{\tau(h)}$ plus noise
```

Loop: for $k = 1 \dots |\tau(h)|$

```
  $t \leftarrow \tau(h)[k]$
  if $t == 1$          // $t$ is the $1^{st}$ timestep
  then for $i \in \{1, 2\}$
    $\delta^h_t(i) \leftarrow \pi_{h_i} O_{h_i}(x_1)$
    $\psi^h_t(i) \leftarrow -1$
```

```
else if $(t-1) == \tau(h)[k-1]$        // the previous timestep is also being optimized
then for $i \in \{1,2\}$
  $\delta_t^h(i) \leftarrow [\max_{j\in\{1,2\}} \delta_{t-1}^h(j) T_{h_j h_i}] O_{h_i}(x_t)$
  $\psi_t^h(i) \leftarrow \arg\max_{j\in\{1,2\}} \delta_{t-1}^h(j) T_{h_j h_i}$
else for $i \in \{1,2\}$        // the previous timestep belongs to a different state $q_{t-1}^*$
  $\delta_t^h(i) \leftarrow T_{q_{t-1}^* h_i} O_{h_i}(x_t)$
```

Termination:

For all subsequences of $\tau(h)$ that are contiguous in $\{1 \ldots \tau\}$, backtrack through ψ^h from the end of the subsequence to its beginning to retrieve the corresponding portion of H_h^*.

return H_h^*.

The running time is clearly linear in the number of non-determined timesteps $|\tau(h)|$ since each maximization in the algorithm is always over 2 elements no matter how large the actual HMM gets. Note that we allow the incoming and outgoing transition parameters of h_1, h_2 to be updated as well, which allows better modeling of dynamic structure during split design.

Split-State Baum-Welch

Split-State Baum-Welch also learns locally optimal values for the state-split parameters λ_{h_1,h_2}, and like Baum-Welch it does so by modeling the *posterior* over the hidden state space which in this case consists of $\{h_1, h_2\}$. The following two steps are repeated for each iteration i until convergence:

1. E-step: Calculate stepwise occupancy and transition probabilities $\{\gamma^h, \xi^h\}$ from $\{\lambda_{\backslash\tau(h)}, \lambda_{h_1,h_2}^i, X_{\tau(h)}, H_{\backslash\tau(h)}^*\}$; compute expectations.

2. M-step:
   $\{\lambda_{h_1,h_2}\}^{i+1} \leftarrow \arg\max_\lambda P(X_{\tau(h)}, H_{\backslash\tau(h)}^* \mid \lambda_{\backslash h}, \lambda)$

The E-step step is carried out using a specialized two-state partially-constrained-path version of the forward-backward algorithm to first calculate the forward and backward variables, which then give the required γ^h and ξ^h values for h_1 and h_2. The idea is the same as Split-State Viterbi but with soft counts and updates. The entire algorithm is $\mathcal{O}(m|\tau(h)|)$, since the update step requires summations over all observations in $\tau(h)$ for at least m transition parameters. Since it is not possible to compute the updated overall likelihood from this split algorithm, convergence of Split-State Baum-Welch is heuristically based on differences in the split-state α variables in successive timesteps.

Though slower, Split-State Baum-Welch searches a larger space of candidate models than Split-State Viterbi Training, performing better on ‘difficult’ splits (such as the one required in Figure 4.1 with high hidden variable entropy) just as Baum-Welch performs better than Viterbi Training in such situations.

4.3.3 Efficient Candidate Scoring and Selection

We compare candidates *amongst each other* using the fast-to-compute Viterbi likelihood after optimizing the split parameters. Afterwards, as we described earlier, we compare the best candidate to *the original model* using BIC score [41] (a.k.a. *Schwarz criterion*), an efficiently computable approximation of the true posterior probability. The latter is intractable since it involves integrating over exponentially many possible models. The BIC score is an asymptotically accurate estimate (in the limit of large data) of the posterior probability of a model when assuming a uniform prior. A Laplace approximation is applied to the intractable integral in the likelihood term, and terms that do not depend on data set size are dropped to obtain the approximated posterior log-probability. BIC effectively punishes complexity by penalizing the number of free parameters, thus safeguarding against overfitting, while rewarding goodness-of-fit via the data log-likelihood. Let $\#\lambda$ denote the number of free parameters in HMM λ. Recall that τ denotes the length of the data sequence, and n denotes the number of discrete observations (or dimensionality of the real-valued observation vector). Then,

$$\mathrm{BIC}(\lambda, X) = \log P(X \mid \lambda) - \frac{\log(\tau n)}{2} \cdot \#\lambda$$

For V-STACS, the likelihood is approximated by the Viterbi path likelihood using the *Viterbi approximation* [42, 43]. The unsplit model Viterbi path can be updated efficiently to compute the Viterbi paths and Viterbi path likelihoods for each candidate in $\mathcal{O}(\tau/m)$ amortized time, and hence $\mathcal{O}(\tau)$ total. This avoids an extra $\mathcal{O}(\tau m^2)$ step in V-STACS, and keeps the complexity of V-STACS' candidate generation, scoring and selection at $\mathcal{O}(m\tau)$.

BIC as defined holds for probabilistic models with observable random variables. More recent work has shown that the effective dimension of *latent variable* Bayesian Networks is equal to the rank of the Jacobian of the transformation between the parameters of the latent variables and the parameters of the observable variables [45]. Using the number of free parameters is a reasonable approximation since this corresponds to the maximum possible rank, and regular BIC with this measure has been successfully used for model selection in HMMs in previous work [37]. Nonetheless, due to the approximate nature of BIC for latent-variable models [45] or for high-dimensional data, test-set log-likelihood would be a more accurate though more computationally expensive scoring criterion for splits.

4.4 Experiments

In our experiments we seek to compare STACS and V-STACS to Li-Biswas, ML-SSS, and multi-restart Baum-Welch, in these areas:

1. Learning models of predetermined size
2. Model selection capability
3. Classification accuracy for multi-class sequential data

For (1) and (2), we are concerned both with the quality of models learned as indicated by test-set likelihoods (for learning predetermined-size models) and BIC scores (for learning state space dimension), as well as running-time efficiency. For (3), we examine sequential data where each sequence is associated with a distinct class label. For such data, HMMs have been successfully used in previous applications (e.g. speech recognition [4]) to construct a *classifier* by training class-specific HMMs on sequences from different classes, and using their likelihood scores on test-set sequences for classification. We will follow the same procedure for the multiclass sequential data we examine.

4.4.1 Algorithms and Data Sets

As described earlier, STACS uses Baum-Welch for parameter learning and Split-State Baum-Welch for split design. V-STACS uses Viterbi Training and Split-State Viterbi Training, followed by Baum-Welch on the final model. We also implemented ML-SSS (with a tweak for generalizing to non-chain topologies) and Li-Biswas for comparison.

We choose a wide range of real-world datasets that have appeared in previous work in various contexts, the goal being to examine the ability of our algorithms to uncover hidden structure in many different, realistic domains. The dimensionality of all datasets was reduced by PCA to 5 for efficiency, except for the Motionlogger dataset which is 2-D. The AUSL and Vowel data sets are from the UCI KDD archive [46]. We list the data sets with (training-set, test-set) sizes below.

Robot: This data set consists of laser-range data provided by the Radish Robotics Data set Repository [47] gathered by a Pioneer indoor robot traversing multiple runs of a closed-loop set of corridors in USC's Salvatori Computer Science building. This data set has appeared in previous work in relation to a robot localization task. Among the four runs of data provided we used three for training ($12,952$ observations) and one for testing ($4,052$ observations).

Mlog: This data set consists of real-valued motion data from two wearable accelerometers worn by a test subject for a period of several days during daily routine. The training set and test set contain $10,000$ and $4,720$ observations respectively. This data set was previously appeared in a paper on large-state-space HMMs [48] and allows us to compare our test-set likelihoods with previous results.

Mocap: This is a small subset of real-valued motion capture data gathered by several

Table 4.1: Test-set log-likelihoods (scaled by dataset size) and training times of HMMs learned using STACS,V-STACS, ML-SSS and regular Baum-Welch with 5 random restarts. The best score and fastest time in each row are highlighted. Li-Biswas had similar results as ML-SSS, and slower running times, for those m where it completed successfully.

Data	STACS	V-STACS	ML-SSS	Baum-Welch	STACS	V-STACS	ML-SSS	Baum-Welch
	$m = 5$				$m = 40$			
ROBOT	-2.41	**-2.41**	-2.44	-2.47	**-1.75**	-1.76	-1.80	-1.78
	40s	***13s***	*70s*	*99s*	*11790s*	***1875s***	*16048s*	*18460s*
MOCAP	-4.46	-4.46	-4.49	-4.46	-4.32	**-4.29**	-4.30	-4.37
	34s	***14s***	*49s*	*65s*	*5474s*	***1053s***	*6430s*	*7315s*
MLOG	-8.78	**-8.78**	-10.49	-8.78	**-8.25**	-8.26	-10.49	-8.38
	67s	*15s*	***9s***	*750s*	*29965s*	*8146s*	***1818s***	*42250s*
AUSL	-3.60	-3.60	-3.60	**-3.43**	-2.89	**-2.77**	-3.08	-2.99
	39s	***14s***	*33s*	*110s*	*7923s*	***1550s***	*8465s*	*22145s*
VOWEL	-4.69	-4.69	-4.68	**-4.67**	-4.34	**-4.32**	-4.44	-4.33
	13s	***8s***	*37s*	*95s*	*2710s*	***1011s***	*2874s*	*6800s*
	$m = 20$				$m = 60$			
ROBOT	**-1.93**	-1.93	-1.98	-1.96	-1.65	**-1.64**	-1.69	-1.75
	2368s	***512s***	*2804s*	*4890s*	*38696s*	***6086s***	*51527s*	*35265s*
MOCAP	-4.38	-4.37	-4.33	-4.33	**-4.23**	-4.26	-4.23	-4.46
	899s	***203s***	*800s*	*3085s*	*16889s*	***3470s***	*18498s*	*20950s*
MLOG	**-8.34**	-8.34	-10.49	-8.40	-8.25	**-8.23**	-8.29	-8.39
	3209s	*1173s*	***284s***	*12350s*	*116891s*	***29379s***	*108358s*	*87150s*
AUSL	-3.16	-3.18	-3.21	**-3.13**	**-2.71**	-2.71	-2.86	-2.89
	1128s	***284s***	*1410s*	*3655s*	*23699s*	***4613s***	*25156s*	*60035s*
VOWEL	**-4.40**	-4.41	-4.44	-4.41	**-4.30**	-4.31	-4.44	-4.31
	548s	***189s***	*1009s*	*1285s*	*8296s*	***2714s***	*4407s*	*13360s*

human subjects instrumented with motion trackers and recorded by cameras while performing sequences of physical movements. It appears in previous work [49] in the context of a classification task on natural and unnatural motion sequences; the best model for the task was found to be an HMM ensemble model and the state spaces required ranged from $50-60$ to 180 states, which makes it a natural candidate for inclusion here. We use a training set of size $10,028$ and test set of size $5,159$.

AUSL: This data set consists of high-quality instrumented glove readings of Australian sign-language words being expressed by an expert signer. The data set contains 27 repetitions each of 95 different words, with each sign consisting of around 50 22-dimensional observations. Here we concatenate signings of 10 different words to form a training set of size $13,435$ and a test set of size $1,771$.

VOWEL: This data set consists of multiple utterances of a particular Japanese vowel by nine male speakers. We broke it up into training and test sets of $4,274$ and $5,687$ datapoints each.

Synthetic data: We generated synthetic data sets to examine the ability of our algorithms to uncover the true number of states.

4.4.2 Learning HMMs of Predetermined Size

We first evaluate performance in learning models of predetermined size. In Table 4.1 we show test-set log-likelihoods normalized by data set size along with running times for experiments using STACS and V-STACS along with ML-SSS and regular Baum-Welch with 5 restarts. Figure 4.4 shows a subset of the same data in a more visually interpretable form for the $m = 40$ case. Here we ignore the stopping criterion and perform the best split at each model selection step until we reach m states. For Baum-Welch, the best score from its five runs is given along with the total time. Li-Biswas results are not shown because most desired model sizes were not reached. However, the instances that did successfully complete indicate that Li-Biswas is much slower than any other method considered, even Baum-Welch, while learning models with similar scores as ML-SSS.

Table 4.2: BIC scores scaled by dataset size, and *(number of states)*, of final models chosen by STACS, V-STACS, Li-Biswas and ML-SSS. STACS and V-STACS consistently find larger models with better BIC scores, indicating more effective split design.

Dataset	STACS	V-STACS	Li-Biswas	ML-SSS
ROBOT	**-1.79*(39)***	-1.81*(34)*	-1.98*(18)*	-2.01*(15)*
MOCAP	**-3.54*(36)***	-3.55*(33)*	-3.69*(20)*	-3.92*(10)*
MLOG	**-8.44*(14)***	-8.45*(20)*	-8.59*(11)*	-10.51*(1)*
AUSL	**-2.77*(44)***	-2.79*(42)*	-2.92*(31)*	-3.04*(28)*
VOWEL	**-4.47*(17)***	-4.49*(16)*	-4.48*(17)*	-4.94*(1)*

We note that STACS and V-STACS have the fastest running times for any given HMM size and data set, except for cases when a competing algorithm got stuck and terminated prematurely. Figure 4.3(A) shows a typical example of STACS and V-STACS running times compared to previous methods for different m values.

As m grows larger and the possibility of local minima increases, STACS and V-STACS consistently return models with better test-set scores. Figure 4.3(B) shows training-set score against running time for the ROBOT data for $m = 40$. This is especially remarkable for V-STACS which is a purely hard-updates algorithm. One possible explanation is that V-STACS' coarseness helps it avoid overfitting when splitting states.

4.4.3 Model Selection Accuracy with BIC

The final BIC scores and m values of STACS, V-STACS, Li-Biswas and ML-SSS are shown in Table 4.2 when allowed to stop splitting autonomously. Note that the exact BIC score was not used by V-STACS, just calculated for comparison. Figure 4.5 presents part of the data in more visually interpretable form. In all cases, STACS converges on models with the highest BIC score. For the MLOG data, STACS achieves a better BIC score even with a smaller model than V-STACS, indicating that the soft-updates method found a particularly good local optimum.

The consistent superiority of STACS here may seem to contradict results from the previous section where STACS and V-STACS were seen to be more comparable. A possible reason is that V-STACS uses the Viterbi path likelihood (which is computed during hard-updates training anyway) in place of the true likelihood in BIC. This is done to keep V-STACS as efficient as possible. However the resulting approximate BIC seems to undervalue good splits, resulting in early stoppage as seen here. We can conclude that, though the Viterbi approximation works well for state-splitting, the true likelihood is preferable for model selection purposes when using BIC.

4.4.4 Discovering the Correct Topology

We already saw that STACS is able to learn the correct number of states in the simple example of Figure 4.1, while Li-Biswas and ML-SSS are not. We generalized this example to a larger, more difficult instance by generating a $10,000$ point synthetic data set (Figure 4.6(A)) similar to the one in Figure 4.1 but with 10 hidden states with overlapping Gaussian observations.

Even on this data, both STACS and V-STACS consistently found the true underlying 10-state model whereas Li-Biswas and ML-SSS could not do so. Interestingly, regular Baum-Welch on a 10-state HMM also failed to find the best configuration of these 10 states even after 50 restarts. This reinforces a notion suggested by results in Section 4.4.2: even in fixed-size HMM learning, STACS is more effective in avoiding local minima than multi-restart Baum-Welch.

4.4.5 Australian Sign-Language Recognition

Though improved test-set likelihood is strong evidence of good models, it is also important to see whether these model improvements translate into superior performance on tasks such as classification. HMMs play one of their most important roles in the context of supervised classification and recognition systems, where one HMM is trained for each

Table 4.3: Australian sign-language word recognition accuracy on a 95-word classification task, and *average HMM sizes*, on AUSL data.

STACS	V-STACS	Li-Biswas	ML-SSS
90.9%	95.8%	78.6%	89.5%
12.5	*55*	*8.3*	*8.5*

distinct sequence class. Classification is carried out by scoring a test sequence with each HMM, and the sequence is labeled with the class of the highest-scoring HMM.

One such classification problem is *automatic sign-language recognition* [50]. We test the effectiveness of our automatically learned HMMs at classification of Australian sign language using the AUSL dataset [1]. The data consists of sensor readings from a pair of Flock instrumented gloves (Figure 4.6(B)), for 27 instances each of 95 distinct words. Each instance is roughly 55 timesteps. We retained the $(x, y, z, roll, pitch, yaw)$ signals from each hand resulting in 12-dimensional sequential data. We trained HMMs on an 8:1 split of the data, using STACS, V-STACS, Li-Biswas and ML-SSS. Table 4.3 shows classification results along with average HMM sizes. V-STACS yields the highest accuracy along with much larger HMMs than the other algorithms.

4.5 Application: Event Detection in Unstructured Audio

Analysis of unstructured audio scene data has a variety of applications such as: *ethno-musicology*, i.e., music classification based on cultural style [51]; *audio diarization*, i.e., extraction of speech segments in long audio signals from background sounds [52]; *audio event detection* [53] for audio mining; *acoustic surveillance* [54], especially for military and public safety applications, e.g, in urban search and rescue scenarios; and *human robot interaction* [55], for voice activated robot actuation and control.

For all these applications, accurate separation of speech from non-speech signals, or background noise, is a fundamental task that can be effectively solved by applying various sequence classification algorithms. One very popular and effective classification scheme [56] is based on HMMs. HMMs can capture the underlying hidden states in the time-series audio data and also model the transitions in these states to represent the underlying dynamical system. Traditionally, HMMs for these applications have been learned by using iterative parameter learning approaches such as Baum-Welch (EM) [4]. While these approaches have had some success, due to limitations of Baum-Welch they have also struggled with issues of computational complexity, reliability and scalability. One reason for this is that Baum-Welch does not aid in the discovery of an appropriate number of states, which is usually done heuristically or exhaustively. Since new data is obtained rapidly and the setting can change over time, HMM training has to be highly efficient for this application, in addition to being accurate. We applied STACS and V-STACS to the task of learning models for detecting a variety of phenomena from raw audio data over time. The data was obtained from MobileFusion, Inc., a Pittsburgh-based company which (at the time this research was undertaken) was developing a commercial audio-event detection software along with an underlying hardware platform. Here we describe the results of this application.

4.5.1 Data and Preprocessing

Figure 4.7(A) shows the portable sensor platform where real-time audio-event detection is to be carried out. For our experiments, audio examples were collected using the device shown in Figure 4.7(B), and were selected with the intent to cover a wide variety of sounds representative of the respective classes of audio events. For instance, the examples produced to support Human class models included instances of male, female and children voices speaking in various languages and environments including offices, coffee shops and urban outdoors. The Animal class examples included sounds of dogs, owls, wolves, coyotes, roosters and birds. The class of Ground Vehicles included sample sounds of motorcycles, cars and trucks passing by. The Aerial Vehicles, our smallest class with just 20 samples, included sounds of jets and helicopters. The Explosions class included sounds of machine-gun fire, grenade blasts and gunshots. The Background class covers other sounds such as ocean waves, city traffic, rain, thunder, crowds plus some silence samples. Details of the data are given in Table 4.4.

Since elaborate, computationally expensive feature extraction methods are not possible during online audio event detection, we applied minimal preprocessing to the raw sound recordings in order to prepare data for audio scene experiments. We extracted a set of 13 Mel-frequency cepstral coefficients (which is efficient) followed by assigning class labels to each of the examples. The cepstral features were then averaged every 5 timesteps to reduce noise and the resulting values were scaled up by a factor of 10 to avoid numerical issues in the multivariate Gaussian code.

Table 4.4: Data Specifications. The last row shows the total number of samples, overall average number of timesteps per sample, and total duration in minutes and seconds.

Class	%	# samples	Av. len. (T)	Duration
Human	27	128	209	40:11
Animal	17	78	37	4:58
Ground V.	17	79	47	6:57
Aerial V.	4	20	54	2:16
Explosion	18	87	11	1:34
Backgrnd.	17	83	21	12:58
	100%	475	78	68:54

4.5.2 Results

The classifier was tested using 4-fold cross-validation on the dataset described above. Figure 4.8 and Table 4.5 summarize the results. We trained HMMs on the data using STACS, V-STACS and EM (Baum-Welch). We used the number of states discovered by V-STACS for EM, which were 56, 26, 26, 18, 14 and 34 on average for the 6 classes respectively. To help EM escape local minima, we re-initialized it 5 times from different random starting points. Training was carried out on a 1.8GHz CPU with 4GB RAM. In Figure 4.8(A), the training times are plotted *in log scale* of minutes, since the disparity between EM and our algorithms was so large. For example, for the Background class, V-STACS took 166 minutes, STACS took 243 minutes, and EM took 1023 minutes, despite having to only learn parameters for a fixed HMM size.

We now look at classification accuracy. We focus on the classification accuracy results of V-STACS vs. EM. STACS results were similar to V-STACS though slightly poorer in some cases. Figure 4.8(B) shows these results for each of the 6 classes, which shows that V-STACS is consistently more accurate than EM, except for labels such as Explosions and Background. We believe this is partly due to lack of sufficient training data (note in Table 4.4 that these classes had the shortest average sample lengths). Table 4.5 shows the confusion matrices for models trained using EM (top) and V-STACS (bottom) respectively. For each pair of actual and predicted labels, the better value of the two tables is listed in **bold** (higher is better on diagonal, lower is better off-diagonal). Due to the highly unstructured, diverse and in some cases insufficient data, neither algorithm is consistently accurate. However V-STACS has better results on the whole, particularly for the Human class which is of particular interest in this application. Some of the false negatives and false positives seen in the confusion matrix can be addressed by using non-uniform priors during classification, to compensate for the large degree of class imbalance.

4.6 Discussion

Part of the contribution of this work is empirical evidence for the conjecture that better modeling of state context compensates more than adequately for considering fewer data points per split in hidden state discovery. In addition, we investigated whether improved dynamic modeling in split design can also compensate for approximating hidden state beliefs by less precise, more efficient hard-updates methods via the V-STACS algorithm. Evaluations show that even V-STACS produces models with higher test-set scores than soft-updates methods like ML-SSS, Li-Biswas and Multi-restart Baum-Welch. The computational efficiency of our methods can make a big difference in a number of practical

Table 4.5: Average Confusion Matrix for EM (top) and V-STACS (bottom). Actual (rows) vs Predicted (columns). Each entry is a percentage of test data averaged over the cross-validation runs, and each row sums to 100. For each entry, the better entry of the two tables is in **bold**. Ties are in *italics*.

	H	A	G.V.	A.V.	E	B
H.	92.69	4.68	**0**	*0*	**0**	2.34
A.	*17.10*	73.68	*0*	1.31	**0**	7.89
G.V.	*9.21*	*3.94*	69.73	2.63	**2.63**	11.84
A.V.	*5*	*0*	15.00	75	**5**	*0*
E.	**9.52**	**1.19**	8.33	**0**	**75**	5.95
B.	15	**8.75**	**5**	**0**	*3.75*	**67.5**
H.	**96.09**	**2.43**	0.78	*0*	0.78	**0**
A.	*17.10*	**81.57**	*0*	**0**	1.31	**0**
G.V.	*9.21*	*3.94*	**72.36**	**0**	3.94	**10.52**
A.V.	*5*	*0*	**5.00**	**80**	10	*0*
E.	14.28	2.38	**4.76**	2.38	72.61	**3.57**
B.	**13.75**	11.25	7.5	**0**	*3.75*	63.75

applications such as audio event-detection (Section 4.5), where frequent on-the-fly retraining is often necessary.

An interesting phenomenon observed is that STACS and V-STACS consistently return *larger* HMMs (with better BIC scores) than ML-SSS and Li-Biswas when stopping autonomously. One possible interpretation is that the split design mechanism continues to find 'good' splits even after the 'easy' splits are exhausted. An illustration of this for the Mocap data is in Figure 4.3.C. This makes sense considering that model selection and parameter optimization are closely related; since parameter search is prone to local minima, determining the best size of a model depends on being able to find regions of parameter space where good candidate models reside. Similarly, it is surprising that that V-STACS yielded the highest classification accuracy in the sign-language recognition task (Section 4.4.5), that too with much larger final HMMs than any other algorithm. V-

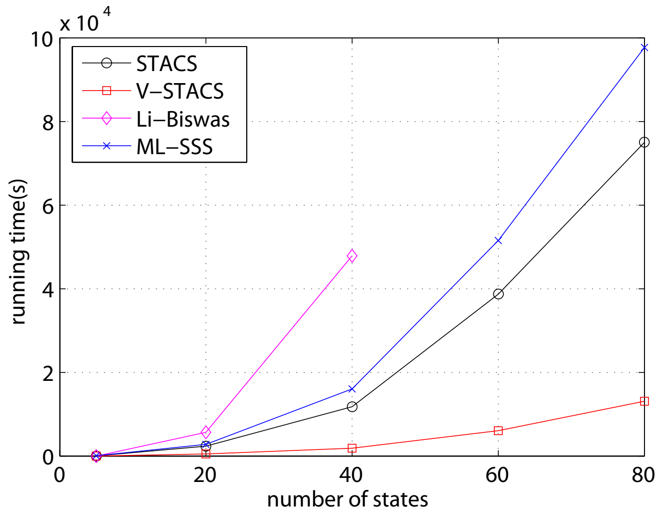
STACS also performed slightly better in the audio event detection task (Section 4.5). More investigation is needed in this area to see if these two things hold true in other sequence classification domains, and if so then why.

Previous work for finding the dimensionality of HMMs with *discrete-valued* observations [42] and other Bayesian Networks with discrete observations [57] has demonstrated that hard-updates model selection algorithms can yield much greater efficiency than soft-updates methods without a large loss of accuracy. To our knowledge, however, this is the first work that demonstrates hard-updates model selection to be competitive for *continuous* observations (Sections 4.4.2, 4.4.5). One possible explanation is that the coarseness of hard-updates splitting helps avoid overfitting early on which might otherwise trap the algorithm in a local optimum. It should also be noted that STACS can be generalized to model selection in Dynamic Bayesian Networks or other directed graphical models that have hidden states of undetermined cardinality, since the sum-product and max-product algorithms for inference and learning in these models are generalizations of the Baum-Welch and Viterbi algorithms for HMMs.

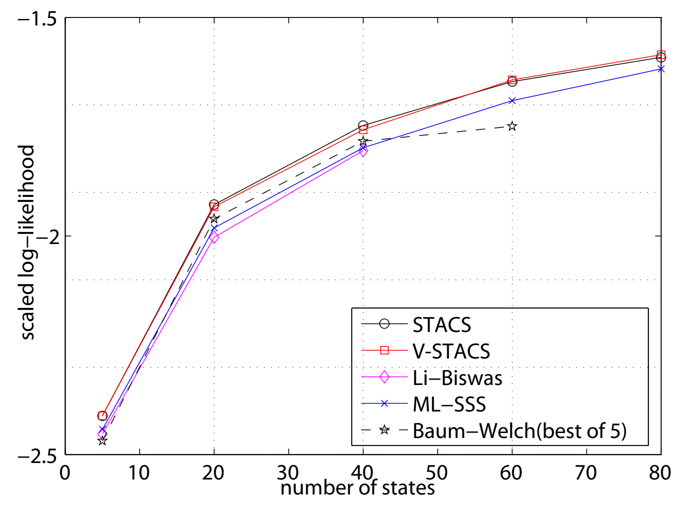
Results from Sections 4.4.2 and 4.4.4 indicate that STACS is also a competitive *fixed-size HMM learning algorithm* compared to previous approaches in terms of test-set scores *and* efficiency. To our knowledge, this is the first HMM model selection algorithm that can make this claim. Consequently, there is great potential for applying STACS to domains where continuous observation-density HMMs of fixed size are used, such as speech recognition, handwriting recognition, financial modeling and bioinformatics.

In the context of this thesis, STACS provides a more efficient way of learning discrete LVMs that also have significantly fewer local-minima issues than those learned using EM. The improved learning algorithm translates to better test-set likelihood and hence increased predictive power in the resulting HMMs.

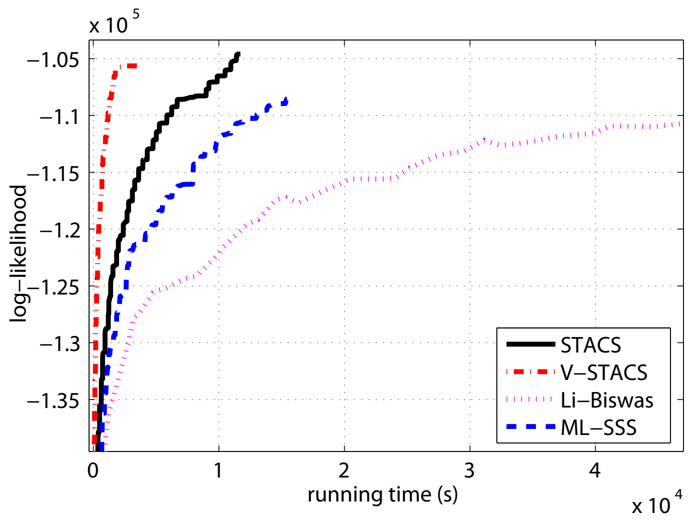
However, though the STACS approach to learning HMMs is strictly better than EM and other existing model selection methods, it does have drawbacks due to its heuristic components. The greedy binary splitting approach could lead to oversplitting in some scenarios, or miss other, better partitionings of the state space. The BIC score is only an approximation to the true posterior, and hence is only an approximate scoring and stopping criterion. Furthermore, regular BIC is not entirely suitable for temporal models [45]. A scoring criterion based on variational Bayes [58] might be a better option, though scoring and stopping based on test-set likelihood, as mentioned earlier, would be best. In Chapter 6 we see a different approach to learning HMMs using matrix decomposition methods. Unlike STACS and EM-based approaches, this algorithm does not suffer from local minima. It also simplifies model selection to a simple examination of singular values obtained from SVD. Such matrix decomposition methods for learning LVMs are more commonly used in learning continuous LVMs in the form of *subspace methods*, which we describe and improve upon in the next chapter.



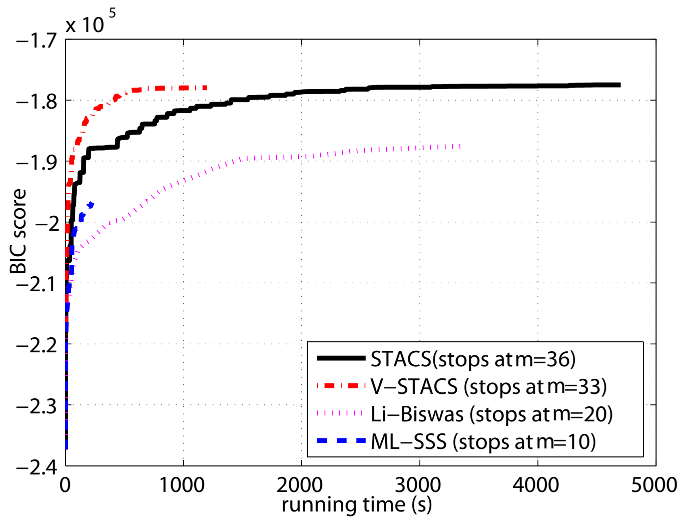
(A)



(B)



(C)



(D)

Figure 4.3: A. Running time vs. number of final states on the ROBOT data set. B. Final scaled log-likelihood (nats/datapoint) vs. number of states for learning fixed-size models on the Robot data. C. Log-Likelihood vs. running time for learning a 40-state model on the ROBOT data. D. BIC score vs. running time on the MOCAP data when allowed to stop splitting autonomously. Results are typical of those obtained on all data sets, shown mostly on the Robot dataset because it allowed the largest $(N = 40)$ Li-Biswas HMMs.

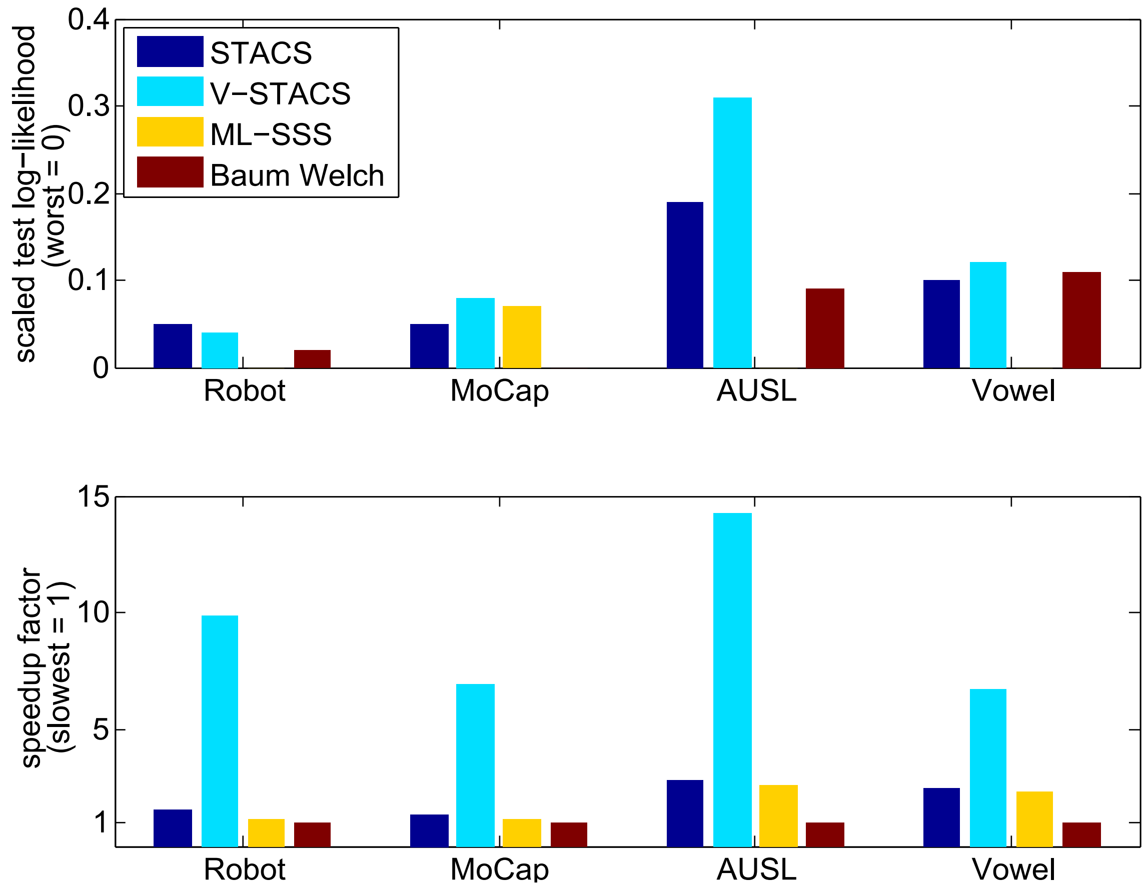


Figure 4.4: Learning 40-state HMMs. Top: scaled test-set loglikelihoods of learned models. Bottom: Factor of running time speedup w.r.t. slowest algorithm for learning.

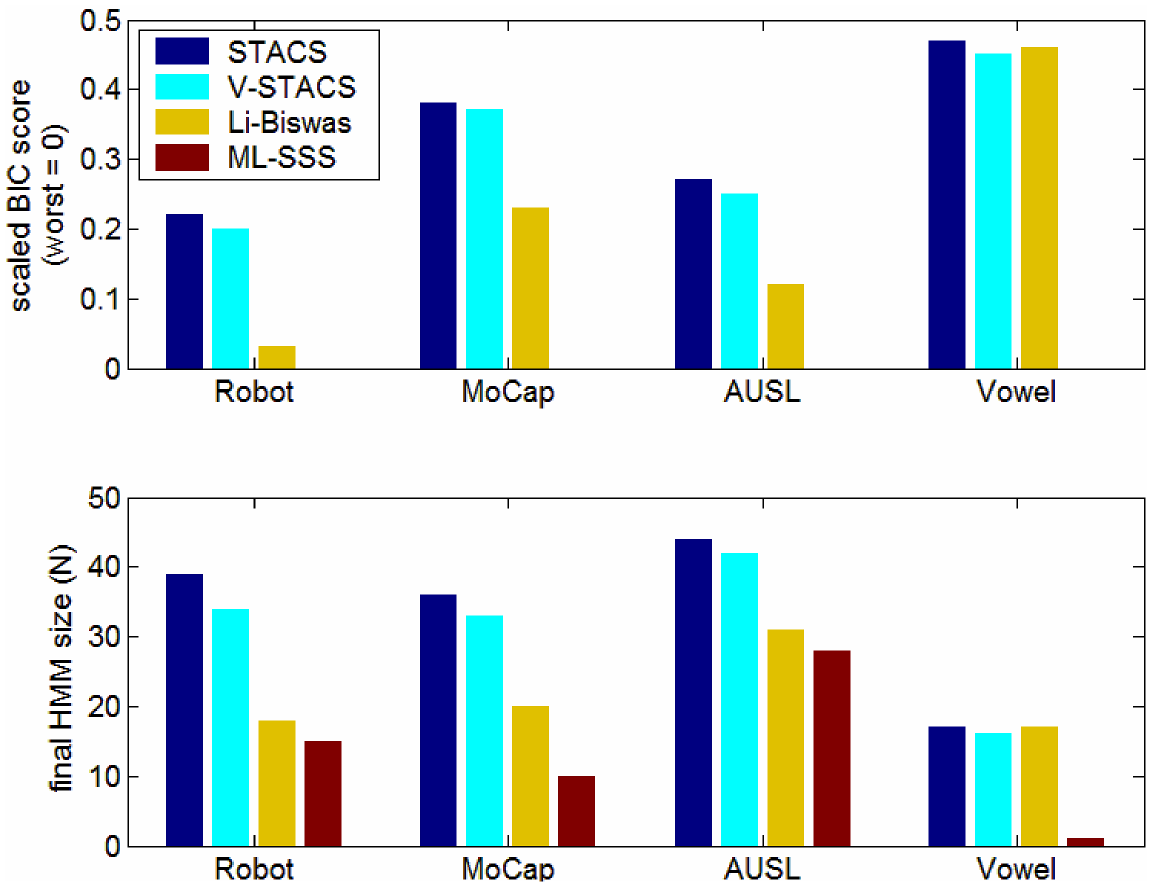
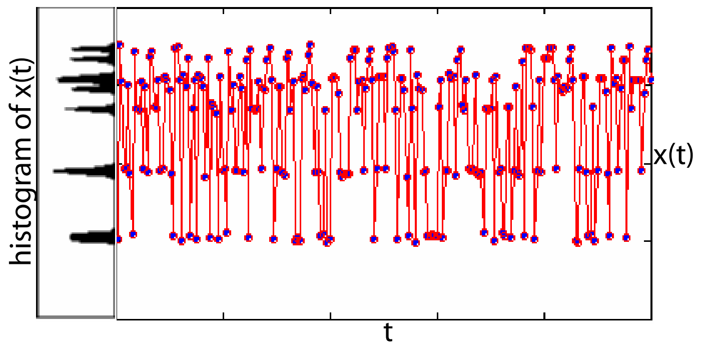


Figure 4.5: Learning state space dimensions and parameters. Top: scaled BIC scores of learned models. Bottom: Final HMM state space size.



(A)

(B)

Figure 4.6: A. Fragment of 10-state univariate synthetic data sequence used for model selection testing. The histogram shows only 7 distinct peaks, indicating that some of the observation densities overlap completely. B. Flock $5DT$ instrumented glove used to collect Australian Sign Language data [1] used in our classification experiments.

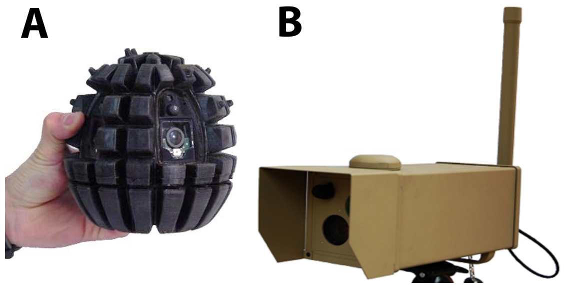
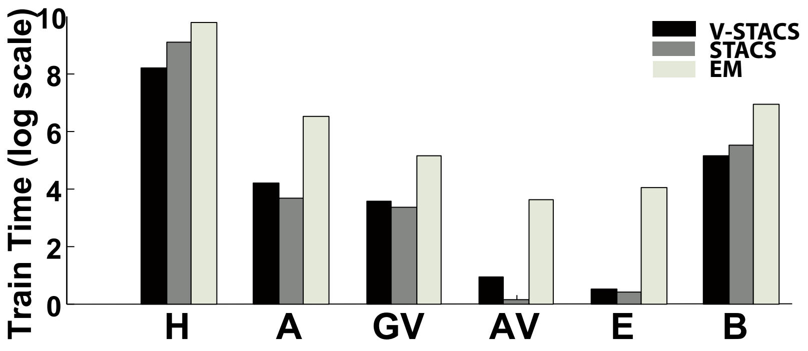
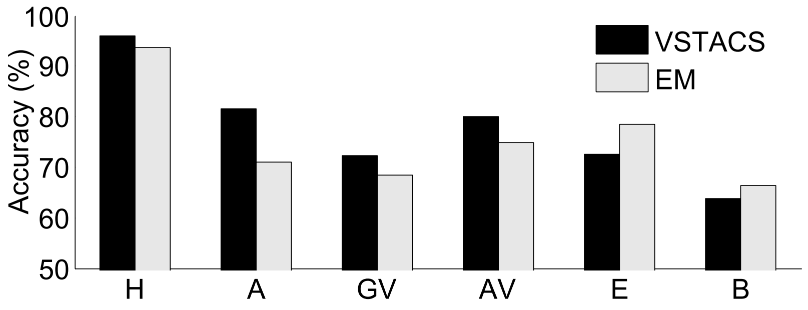


Figure 4.7: (A) A mobile tactical device and (B) a fixed device on which our algorithms were deployed



(A)



(B)

Figure 4.8: (A) Training times *in log(minutes)*. V-STACS and STACS are at least an order of magnitude faster than 5-restart EM. (B) Classification accuracies. VSTACS is better than EM in nearly all cases.

Chapter 5

Learning Stable Linear Dynamical Systems

In this chapter we shift our focus from discrete to continuous latent variable models. We extend Subspace Identification[1] (Section 3.3.2), a popular alternative to the EM algorithm for learning LDS parameters which originates in the controls literature. EM is *statistically efficient*: it promises to find an optimum point of the observed data likelihood for finite amounts of training data. On the downside, however, EM is *computationally inefficient*: each run finds only a *local* optimum of this likelihood function, and hence we need many random re-initializations to search parameter space for the global optimum. Subspace ID reformulates the search space by trading off a small amount of *statistical* efficiency in return for a large increase in *computational* efficiency. Though Subspace ID does not reach an optimum point for finite data samples, it promises to reach the *global* optimum of the observed data likelihood in the limit of sufficient data. Subspace ID is simpler and easier to implement than EM as well, consisting of a singular value decomposition (SVD) of a matrix in contrast to the repeated forward-backward iterations of EM.

An additional difficulty in learning LDSs as opposed to HMMs is that standard learning algorithms can result in models with *unstable* dynamics, which causes them to be ill-suited

[1]though our extension applies to the EM algorithm too, as we describe later

for several important tasks such as *simulation* and *long-term prediction*. This problem can arise even when the underlying dynamical system emitting the data is stable, particularly if insufficient training data is available. which is often the case for high-dimensional temporal sequences. In this chapter we propose an extension to Subspace ID that enforces the estimated parameters to be stable. Though stability is a non-convex constraint, we will see how a constraint-generation-based optimization approach yields approximations to the optimal solution that are more efficient and more accurate than previous state-of-the-art stabilizing methods.

5.1 Introduction

We propose an optimization algorithm for learning the dynamics matrix of an LDS while guaranteeing stability. We first obtain an estimate of the underlying state sequence using subspace identification. We then formulate the least-squares minimization problem for the dynamics matrix as a *quadratic program* (QP) [59], initially without constraints. When we solve this QP, the estimate $\widehat{A}$ we obtain may be unstable. However, any unstable solution allows us to derive a linear constraint which we then add to our original QP and re-solve. This constraint is a conservative approximation to the true feasible region. The above two steps are iterated until we reach a stable solution, which is then refined by a simple interpolation to obtain the best possible stable estimate. The overall algorithm is illustrated in Figure 5.1(A).

Our method can be viewed as *constraint generation* [60] for an underlying convex program with a feasible set of all matrices with singular values at most 1, similar to work in control systems such as [2]. This convex set approximates the true, non-convex feasible region. So, we terminate *before* reaching feasibility in the convex program, by checking for matrix stability after each new constraint. This makes our algorithm less conservative than previous methods for enforcing stability since it chooses the best of a larger set of stable dynamics matrices. The difference in the resulting stable systems is noticeable when simulating data. The constraint generation approach also results in much greater efficiency than previous methods in nearly all cases.

One application of LDSs in computer vision is learning *dynamic textures* from video data [61]. An advantage of learning dynamic textures is the ability to play back a realistic-looking generated sequence of desired duration. In practice, however, videos synthesized from dynamic texture models can quickly become degenerate because of instability in the underlying LDS, or because of the competitive inhibition problems discussed in Chapter 1. In contrast, sequences generated from dynamic textures learned by our method remain "sane" even after arbitrarily long durations, although we leave the problem of *competitive inhibition* to Chapter 6. We also apply our algorithm to learning baseline dynamic models of over-the-counter (OTC) drug sales for biosurveillance, and sunspot numbers from the UCR archive [62]. Comparison to the best alternative methods [2, 63] on these problems yields positive results.

5.2 Related Work

Linear system identification is a well-studied subject [26]. Within this area, *subspace identification methods* [27] have been very successful. These techniques first estimate the model dimensionality and the underlying state sequence, and then derive parameter estimates using least squares. Within subspace methods, techniques have been developed to enforce stability by augmenting the extended observability matrix with zeros [64] or adding a regularization term to the least squares objective [65].

All previous methods were outperformed by Lacy and Bernstein [2], henceforth referred to as LB-1. They formulate the problem as a semidefinite program (SDP) whose objective minimizes the state sequence reconstruction error, and whose constraint bounds the largest singular value by 1. This convex constraint is obtained from the nonlinear matrix inequality $I_n - AA^{\mathsf{T}} \succeq 0$, where I_n is the $n \times n$ identity matrix and $\succ 0$ ($\succeq 0$) denotes positive (semi-) definiteness. This can be seen as follows: the inequality bounds the top singular value by 1 since it implies for all vectors $x \in \mathbb{R}^n$:

$$\begin{aligned} x^{\mathsf{T}}(I_n - AA^{\mathsf{T}})x &\geq 0 \\ \Rightarrow x^{\mathsf{T}}AA^{\mathsf{T}}x &\leq x^{\mathsf{T}}x \end{aligned}$$

Therefore this statement holds for $\nu = \nu_1(AA^\mathsf{T})$. Define $\lambda = \lambda_1(AA^\mathsf{T})$. Then,

$$\begin{aligned}
\nu^T AA^\mathsf{T}\nu \leq \nu^\mathsf{T}\nu \\
\Rightarrow \nu^\mathsf{T}\lambda\nu \leq 1 \\
\Rightarrow \sigma_1^2(A) \leq 1
\end{aligned}$$

where the last step follows from the fact that $\nu^\mathsf{T}\nu = 1$ and $\sigma_1^2(M) = \lambda_1(MM^\mathsf{T})$ for any square matrix M. The existence of this constraint also proves the convexity of the $\sigma_1 \leq 1$ region. This condition is *sufficient* but not *necessary* for stability, since a matrix that violates this condition may still be stable.

A follow-up to this work by the same authors [63], which we will call LB-2, attempts to overcome the conservativeness of LB-1 by approximating the Lyapunov inequalities $P - APA^\mathsf{T} \succ 0$, $P \succ 0$ with the inequalities $P - APA^\mathsf{T} - \delta I_n \succeq 0$, $P - \delta I_n \succeq 0$, $\delta > 0$. These inequalities hold iff the spectral radius is less than 1.[2] However, the approximation is achieved only at the cost of inducing a nonlinear distortion of the objective function by a problem-dependent reweighting matrix involving P, which is a variable to be optimized. In our experiments, this causes LB-2 to perform worse than LB-1 (for any δ) in terms of the state sequence reconstruction error, even while obtaining solutions outside the feasible region of LB-1. Consequently, we focus on LB-1 in our conceptual and qualitative comparisons as it is the strongest baseline available. However, LB-2 is more scalable than LB-1, so quantitative results are presented for both.

To summarize the distinction between LB-1 and LB-2: it is hard to have both the right objective function (reconstruction error) and the right feasible region (the set of stable matrices). LB-1 optimizes the right objective but over the wrong feasible region (the set of matrices with $\sigma_1 \leq 1$). LB-2 has a feasible region close to the right one, but at the cost of distorting its objective function to an extent that it fares worse than LB-1 in nearly all cases.

[2]For a proof sketch, see [32] pg. 410.

5.3 The Algorithm

The overall algorithm we propose is quite simple. We first formulate the dynamics matrix learning problem as a QP with a feasible set that includes the set of stable dynamics matrices. Then, if the unconstrained solution is unstable, we demonstrate how unstable solutions can be used to *generate linear constraints* that are added to restrict the feasible set of the QP appropriately. The QP is then re-solved and the constraint generation loop is repeated until we reach a stable solution. As a final step, the solution is refined to be as close as possible to the unconstrained-objective-minimizing estimate while remaining stable. The overall algorithm is illustrated in Figure 5.1(A). Note that the linear constraints can eliminate subsets of the set of stable matrices from the solution space, so the final solution is not necessarily the optimal one. However, with respect to LB-1and LB-2, our method optimizes the right objective (unlike LB-2) over a less conservative feasible region which includes some stable matrices with $\sigma_1 > 1$ (unlike LB-1). Optimizing over the right feasible region (spectral radius ≤ 1) is hard, for reasons we will see in Section 5.3.2.

We now elaborate on the different steps of the algorithm, namely how to formulate the objective (Section 5.3.1), generate constraints (Section 5.3.3), compute a stable solution (Section 5.3.4) and then refine it (Section 5.3.5).

5.3.1 Formulating the Objective

Assume the notation and formulations of Chapter 3. In subspace ID as well as in the M-step of an iteration of EM, it is possible to write the objective function for $\widehat{A}$ as a quadratic function. For subspace ID we define a quadratic objective function:

$$\begin{aligned}
\widehat{A} &= \arg\min_{A} \left\| A\widetilde{X}_i - \widetilde{X}_{i+1} \right\|_F^2 \\
&= \arg\min_{A} \left\{ \operatorname{tr}\left[\left(A\widetilde{X}_i - \widetilde{X}_{i+1} \right)^{\mathsf{T}} \left(A\widetilde{X}_i - \widetilde{X}_{i+1} \right) \right] \right\} \\
&= \arg\min_{A} \left\{ \operatorname{tr}\left(A\widetilde{X}_i \widetilde{X}_i^{\mathsf{T}} A^{\mathsf{T}} \right) - 2\operatorname{tr}\left(\widetilde{X}_i \widetilde{X}_{i+1}^{\mathsf{T}} A \right) + \operatorname{tr}\left(\widetilde{X}_{i+1}^{\mathsf{T}} \widetilde{X}_{i+1} \right) \right\} \\
&= \arg\min_{a} \left\{ a^{\mathsf{T}} P a - 2\, q^{\mathsf{T}} a + r \right\} \qquad (5.1a)
\end{aligned}$$

where $a \in \mathbb{R}^{n^2 \times 1}$, $q \in \mathbb{R}^{n^2 \times 1}$, $P \in \mathbb{R}^{n^2 \times n^2}$ and $r \in \mathbb{R}$ are defined as:

$$a = \text{vec}(A) = [A_{11}\ A_{21}\ A_{31}\ \cdots\ A_{nn}]^\mathsf{T} \tag{5.1b}$$

$$P = I_n \otimes \left(\widetilde{X}_i \widetilde{X}_i^\mathsf{T}\right) \tag{5.1c}$$

$$q = \text{vec}(\widetilde{X}_i \widetilde{X}_{i+1}^\mathsf{T}) \tag{5.1d}$$

$$r = \text{tr}\left(\widetilde{X}_{i+1}^\mathsf{T} \widetilde{X}_{i+1}\right) \tag{5.1e}$$

I_n is the $n \times n$ identity matrix and $\otimes$ denotes the Kronecker product. Note that P is a symmetric positive semi-definite matrix and the objective function in Equation (5.1a) is a quadratic function of a. For EM, we can use a similar quadratic objective function:

$$\widehat{A} = \arg\min_a \left\{a^\mathsf{T} P a - 2\, q^\mathsf{T} a\right\} \tag{5.2a}$$

where $a \in \mathbb{R}^{n^2 \times 1}$, $q \in \mathbb{R}^{n^2 \times 1}$ and $P \in \mathbb{R}^{n^2 \times n^2}$ are defined as:

$$a = \text{vec}(A) = [A_{11}\ A_{21}\ A_{31}\ \cdots\ A_{nn}]^\mathsf{T} \tag{5.2b}$$

$$P = I_n \otimes \left(\sum_{t=2}^{T} P_t\right) \tag{5.2c}$$

$$q = \text{vec}\left(\sum_{t=2}^{T} P_{t-1,t}\right) \tag{5.2d}$$

Here, P_t and $P_{t-1,t}$ are taken directly from the E-step of EM.

5.3.2 Convexity

The feasible set of the quadratic objective function is the space of all $n \times n$ matrices, regardless of their stability. When its solution yields an unstable matrix, the spectral radius of $\widehat{A}$ is greater than 1. Ideally we want to constrain the solution space to the set of stable matrices, i.e. the *set of matrices with spectral radius at most one* (which we call S_λ). However, it is not possible to formulate a convex optimization routine that optimizes over this set because of the shape of S_λ. Consider the class of 2×2 matrices [66]: $E_{\alpha,\beta} = [\,0.3\ \ \alpha\,;\beta\ \ 0.3\,]$. The matrices $E_{10,0}$ and $E_{0,10}$ are stable with $\lambda_1 = 0.3$, but their convex

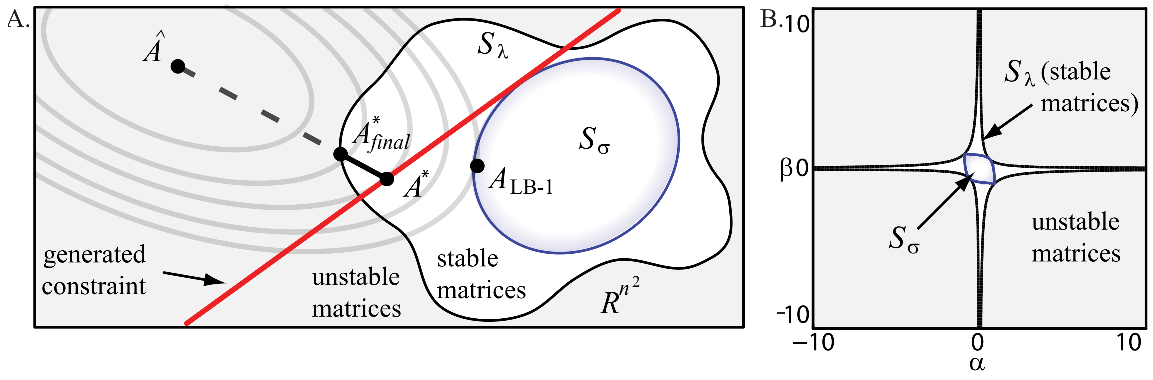


Figure 5.1: (A): Conceptual depiction of the space of $n \times n$ matrices. The region of stability (S_λ) is non-convex while the smaller region of matrices with $\sigma_1 \leq 1$ (S_σ) is convex. The elliptical contours indicate level sets of the quadratic objective function of the QP. $\widehat{A}$ is the unconstrained least-squares solution to this objective. $A_{\text{LB-1}}$ is the solution found by LB-1 [2]. One iteration of constraint generation yields the constraint indicated by the line labeled 'generated constraint', and (in this case) leads to a stable solution A^*. The final step of our algorithm improves on this solution by interpolating A^* with the previous solution (in this case, $\widehat{A}$) to obtain A^*_{final}. (B): The actual stable and unstable regions for the space of 2×2 matrices $E_{\alpha,\beta} = [\,0.3 \;\; \alpha\,;\beta \;\; 0.3\,]$, with $\alpha, \beta \in [-10, 10]$. Constraint generation is able to learn a nearly optimal model from a noisy state sequence of length 7 simulated from $E_{0,10}$, with better state reconstruction error than either LB-1 or LB-2.

combination $\gamma E_{10,0} + (1-\gamma)E_{0,10}$ is unstable for (e.g.) $\gamma = 0.5$ (Figure 5.1(B)). This shows that the set of stable matrices is non-convex for $n = 2$, and in fact this is true for all $n > 1$. The problem of optimizing over this set is hence a difficult non-convex optimization routine. We turn instead to the largest *singular value*, which is a closely related quantity since

$$\sigma_{min}(\widehat{A}) \leq |\lambda_i(\widehat{A})| \leq \sigma_{max}(\widehat{A}) \quad \forall i = 1, \ldots, n \quad [32]$$

Therefore every unstable matrix has a singular value greater than one, but the converse is not necessarily true. Moreover, the set of matrices with $\sigma_1 \leq 1$ *is* convex. To see this, note that for any square matrix M,

$$\sigma_1(M) \equiv \max_{u,v:\|u\|_2=1,\|v\|_2=1} u^\mathsf{T} M v.$$

Therefore, if $\sigma_1(M_1) \leq 1$ and $\sigma_1(M_2) \leq 1$, then for all convex combinations,

$$\sigma_1(\gamma M_1 + (1-\gamma)M_2) = \max_{u,v:\|u\|_2=1,\|v\|_2=1} \gamma u^\mathsf{T} M_1 v + (1-\gamma) u^\mathsf{T} M_2 v \leq 1.$$

Figure 5.1(A) conceptually depicts the non-convex region of stability S_λ and the convex region S_σ with $\sigma_1 \leq 1$ in the space of all $n \times n$ matrices for some fixed n. The difference between S_σ and S_λ can be significant. Figure 5.1(B) depicts these regions for $E_{\alpha,\beta}$ with $\alpha, \beta \in [-10, 10]$. The stable matrices $E_{10,0}$ and $E_{0,10}$ reside at the edges of the figure. Our algorithm is designed to mitigate the difference between S_λ and S_σ by stopping before it reaches S_σ. While one might worry that the difference is too severe to mitigate this way, and results do vary based on the instance used, our experiments below will show that our constraint generation algorithm described below is able to learn a nearly optimal model from a noisy state sequence of $\tau = 7$ simulated from $E_{0,10}$, with better state reconstruction error than LB-1 and LB-2.

5.3.3 Generating Constraints

We now describe how to generate convex constraints on the set S_σ. The basic idea is to use the unstable solution $\widehat{A}$ along with properties of matrices in the set S_σ to infer a linear constraint between $\widehat{A}$ and S_σ. Assume that

$$\widehat{A} = \widetilde{U}\widetilde{\Sigma}\widetilde{V}^\mathsf{T}$$

by SVD, where $\widetilde{U} = [\widetilde{u}_i]_{i=1}^n$, $\widetilde{V} = [\widetilde{v}_i]_{i=1}^n$ and $\widetilde{\Sigma} = \text{diag}\{\tilde{\sigma}_1, \ldots, \tilde{\sigma}_n\}$. Then:

$$\widehat{A} = \widetilde{U}\widetilde{\Sigma}\widetilde{V}^\mathsf{T} \Rightarrow \quad \widetilde{\Sigma} = \widetilde{U}^\mathsf{T}\widehat{A}\widetilde{V} \Rightarrow \quad \tilde{\sigma}_1(\widehat{A}) = \widetilde{u}_1^\mathsf{T}\widehat{A}\widetilde{v}_1 = \text{tr}(\widetilde{u}_1^\mathsf{T}\widehat{A}\widetilde{v}_1) \tag{5.3}$$

Therefore, instability of $\widehat{A}$ implies that:

$$\tilde{\sigma}_1 > 1 \Rightarrow \quad \text{tr}\left(\widetilde{u}_1^{\mathsf{T}} \widehat{A} \widetilde{v}_1\right) > 1 \Rightarrow \quad \text{tr}\left(\widetilde{v}_1 \widetilde{u}_1^{\mathsf{T}} \widehat{A}\right) > 1 \Rightarrow \quad g^{\mathsf{T}} \widehat{a} > 1 \tag{5.4}$$

Here $g = \text{vec}(\widetilde{u}_1 \widetilde{v}_1^{\mathsf{T}})$. Since equation (5.4) arises from an unstable solution of equation (5.1a), g is can be interpreted as a hyperplane separating $\widehat{a}$ from the space of matrices with $\sigma_1 \leq 1$. In fact, the hyperplane is *tangent* to $\S_{\sigma}$. We use the negation of equation (5.4) as a constraint:

$$g^{\mathsf{T}} \widehat{a} \leq 1 \tag{5.5}$$

5.3.4 Computing the Solution

Given the above mechanism for generating convex constraints on the set S_{σ}, a *constraint generation-based* convex optimization algorithm immediately suggests itself. The overall quadratic program can be stated as:

$$\begin{aligned} &\text{minimize} \quad a^{\mathsf{T}} P a - 2\, q^{\mathsf{T}} a + r \\ &\text{subject to} \quad G a \leq h \end{aligned} \tag{5.6}$$

with a, P, q and r as defined in Eqs. (5.1e). $\{G, h\}$ define the set of constraints, and are initially empty. The QP is invoked repeatedly until the stable region, i.e. S_{λ}, is reached. At each iteration, we calculate a linear constraint of the form in Eq. (5.5), add the corresponding g^{T} as a row in G, and augment the h vector with a 1 at the end. Note that we will almost always stop *before* reaching the feasible region S_{σ}.

5.3.5 Refinement

Once a stable matrix is obtained, it is possible to refine this solution. We know that the last constraint caused our solution to cross the boundary of S_{λ}, so we interpolate between the last solution and the previous iteration's solution using binary search to look for a boundary of the stable region, in order to obtain a better objective value while remaining stable. This results in a stable matrix with top eigenvalue equal to 1. In principle, we could attempt to interpolate between any stable solution and any one of the unstable solutions from previous iterations. However, the stable region can be highly complex, and there may be several folds and boundaries of the stable region in the interpolated area. In our experiments (not shown), interpolating from the Lacy-Bernstein solution to the last unstable solution yielded worse results. We also tried other interpolation and constraint-relaxation methods such as: interpolating from the least squares solution to the first stable solution, dropping constraints added earlier in the constraint-generation process, and expanding the constrained set by multiplying the h vector by a constant greater than one. All these methods yielded worse results overall than the algorithm presented here.

5.4 Experiments

For learning the dynamics matrix, we implemented EM, subspace identification, constraint generation (using `quadprog`), LB-1 [2] and LB-2 [63] (using `CVX` with `SeDuMi`) in Matlab on a 3.2 GHz Pentium with 2 GB RAM. Note that the algorithms that constrain the solution to be stable give a different result from the basic EM and and subspace ID algorithms only in situations when the unconstrained $\widehat{A}$ is unstable. However, LDSs learned in scarce-data scenarios are unstable for almost any domain, and some domains lead to unstable models up to the limit of available data (e.g. the `steam` dynamic textures in Section 5.4.1). The goals of our experiments are to: (1) examine the state evolution and simulated observations of models learned using constraint generation, and compare them to previous work on learning stable dynamical systems; and (2) compare the algorithms in terms of computational efficiency. We apply these algorithms to learning dynamic textures from the vision domain (Section 5.4.1), modeling over-the-counter (OTC) drug sales counts (Section 5.4.3) and sunspot numbers (Section 5.4.4).

5.4.1 Stable Dynamic Textures

Dynamic textures in vision can intuitively be described as models for sequences of images that exhibit some form of low-dimensional structure and recurrent (though not necessarily

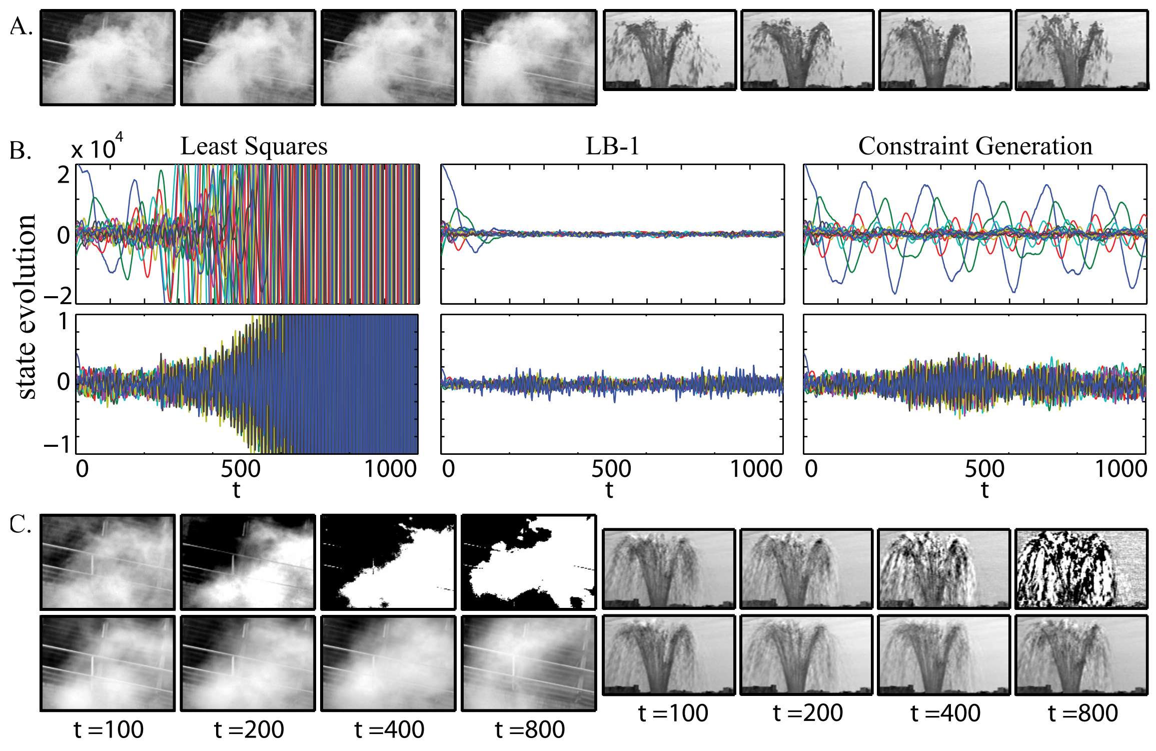


Figure 5.2: Dynamic textures. A. Samples from the original `steam` sequence and the `fountain` sequence. B. State evolution of synthesized sequences over 1000 frames (`steam` top, `fountain` bottom). The least squares solutions display instability as time progresses. The solutions obtained using LB-1 remain stable for the full 1000 frame image sequence. The constraint generation solutions, however, yield state sequences that are stable over the full 1000 frame image sequence without significant damping. C. Samples drawn from a least squares synthesized sequences (top), and samples drawn from a constraint generation synthesized sequence (bottom). The constraint generation synthesized `steam` sequence is qualitatively better looking than the `steam` sequence generated by LB-1, although there is little qualitative difference between the two synthesized `fountain` sequences.

repeating) characteristics, e.g. fixed-background videos of rising smoke or flowing water. Treating each frame of a video as an observation vector of pixel values y_t, we learned

	CG	LB-1	LB-1*	LB-2	CG	LB-1	LB-1*	LB-2
	`steam` $(n = 10)$				`fountain` $(n = 10)$			
$\lvert\lambda_1\rvert$	1.000	0.993	0.993	1.000	0.999	0.987	0.987	0.997
σ_1	1.036	1.000	1.000	1.034	1.051	1.000	1.000	1.054
$e_x(\%)$	**45.2**	103.3	103.3	546.9	**0.1**	4.1	4.1	3.0
time	**0.45**	95.87	3.77	0.50	**0.15**	15.43	1.09	0.49
	`steam` $(n = 20)$				`fountain` $(n = 20)$			
$\lvert\lambda_1\rvert$	0.999	—	0.990	0.999	0.999	—	0.988	0.996
σ_1	1.037	—	1.000	1.062	1.054	—	1.000	1.056
$e_x(\%)$	**58.4**	—	154.7	294.8	**1.2**	—	5.0	22.3
time	**2.37**	—	1259.6	33.55	**1.63**	—	159.85	5.13
	`steam` $(n = 30)$				`fountain` $(n = 30)$			
$\lvert\lambda_1\rvert$	1.000	—	0.988	1.000	1.000	—	0.993	0.998
σ_1	1.054	—	1.000	1.130	1.030	—	1.000	1.179
$e_x(\%)$	**63.0**	—	341.3	631.5	**13.3**	—	14.9	104.8
time	**8.72**	—	23978.9	62.44	**12.68**	—	5038.94	48.55
	`steam` $(n = 40)$				`fountain` $(n = 40)$			
$\lvert\lambda_1\rvert$	1.000	—	0.989	1.000	1.000	—	0.991	1.000
σ_1	1.120	—	1.000	1.128	1.034	—	1.000	1.172
$e_x(\%)$	**20.24**	—	282.7	768.5	**3.3**	—	4.8	21.5
time	**5.85**	—	79516.98	289.79	**61.9**	—	43457.77	239.53

Table 5.1: Quantitative results on the dynamic textures data for different numbers of states n. CG is our algorithm, LB-1and LB-2 are competing algorithms, and LB-1* is a simulation of LB-1 using our algorithm by generating constraints until we reach S_σ, since LB-1 failed for $n > 10$ due to memory limits. e_x is percent difference in squared reconstruction error. Constraint generation, in all cases, has lower error and faster runtime.

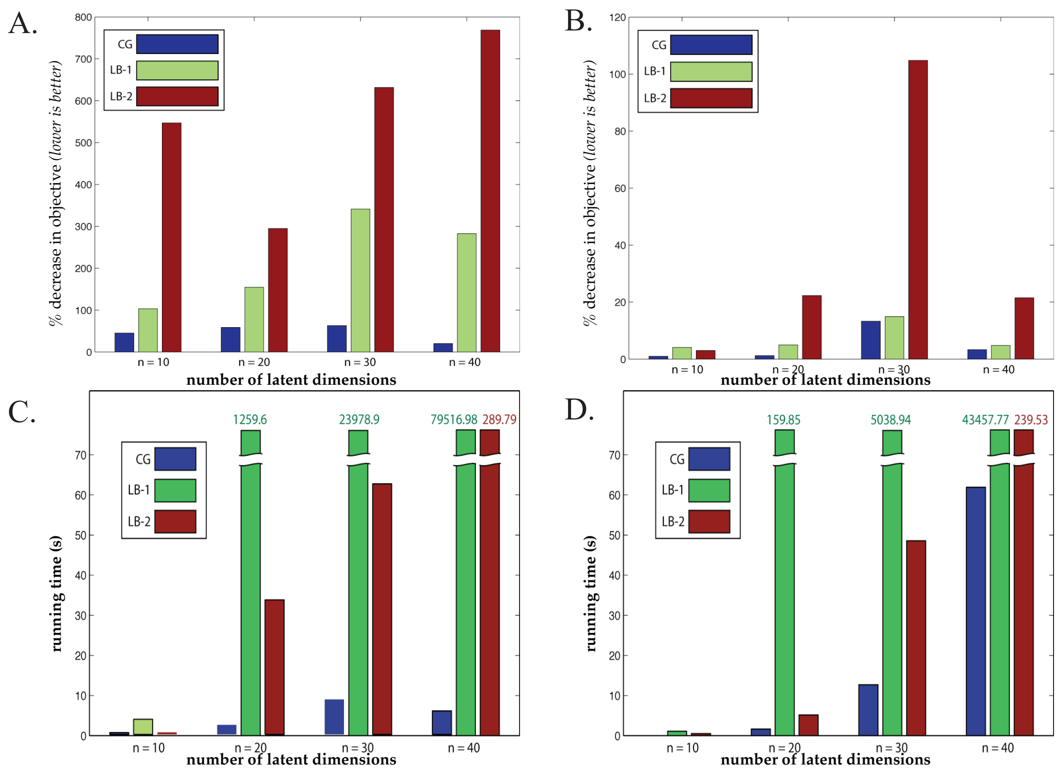


Figure 5.3: Bar graphs illustrating decreases in objective function value relative to the least squares solution (A,B) and the running times (C,D) for different stable LDS learning algorithms on the `fountain`(A,C) and `steam`(B,D) textures respectively, based on the corresponding columns of Table 5.1.

dynamic texture models of two video sequences: the `steam` sequence, composed of 120×170 pixel images, and the `fountain` sequence, composed of 150×90 pixel images, both of which originated from the MIT temporal texture database (Figure 5.2(A)). We use parameters $\tau = 80$, $n = 15$, and $d = 10$. Note that, while the observations are the raw pixel values, the underlying state sequence we learn has no *a priori* interpretation.

An LDS model of a dynamic texture may *synthesize* an "infinitely" long sequence of images by driving the model with zero mean Gaussian noise. Each of our two models uses an 80 frame training sequence to generate 1000 sequential images in this way. To better visualize the difference between image sequences generated by least-squares, LB-1, and constraint generation, the evolution of each method's state is plotted over the course of the synthesized sequences (Figure 5.2(B)). Sequences generated by the least squares models appear to be unstable, and this was in fact the case; both the `steam` and the `fountain` sequences resulted in unstable dynamics matrices. Conversely, the constrained subspace identification algorithms all produced well-behaved sequences of states and stable dynamics matrices (Table 5.1), although constraint generation demonstrates the fastest runtime, best scalability, and lowest error of any stability-enforcing approach.

A qualitative comparison of images generated by constraint generation and least squares (Figure 5.2(C)) indicates the effect of instability in synthesized sequences generated from dynamic texture models. While the unstable least-squares model demonstrates a dramatic and unrealistic increase in image contrast over time, the constraint generation model continues to generate qualitatively reasonable images. Qualitative comparisons between constraint generation and LB-1 indicate that constraint generation learns models that generate more natural-looking video sequences[3] than LB-1.

Table 5.1 demonstrates that constraint generation always has the lowest error as well as the fastest runtime. The running time of constraint generation depends on the number of constraints needed to reach a stable solution. Note that LB-1 is more efficient and scalable when simulated using constraint generation (by adding constraints until S_σ is reached) than it is in its original SDP formulation.

Figure 5.3 shows bar graphs comparing reconstruction errors and running times of these algorithms, based on columns of Table 5.1, illustrating the large difference in efficiency and accuracy between constraint generation and competing methods.

5.4.2 Prediction Accuracy on Robot Sensor Data

Another important measure of accuracy of dynamic models is their performance on short-term and long-term prediction. This problem was addressed in the context of stable LDS modeling in Byron Boots' CMU Masters thesis [3], which uses the techniques described

[3]See videos at http://www.select.cs.cmu.edu/projects/stableLDS

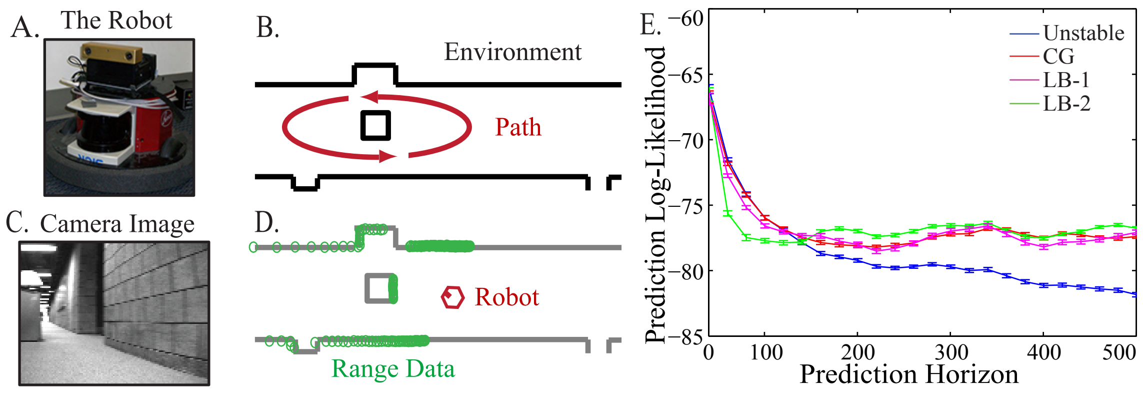


Figure 5.4: Prediction accuracy on Robot Sensory Data (from Boots (2009) [3]). A. The mobile robot with camera and laser sensors. B. The environment and robot path. C. An image from the robot camera. D. A depiction of laser range scan data (green dots on environment surfaces). E. Predictive log-likelihoods from: the unstable model (Unstable), constraint generation (CG), and the two other LDS stabilizing algorithms, LB-1 and LB-2. Bars at every 20 timesteps denote variance in the results. CG provides the best stable short term predictions, nearly mirroring the unstable model, while all three stabilized models do better than the unstable model in the long term.

in this chapter for models with exogenous control inputs, and also combines the constraint generation algorithm with EM (Section 3.3.1) in the way indicated earlier in this chapter. The experiment and result is summarized in this section since the original document is difficult to access outside CMU. Vision data and laser range scans were collected from a Point Grey Bumblebee2 stereo camera and a SICK laser rangefinder mounted on a Botrics O-bot d100 mobile robot platform (Figure 5.4(A)) circling an obstacle in an indoor environment(Figure 5.4(B)). After collecting video and laser data (Figure 5.4(C,D)), the pixel and range reading data were vectorized at each timestep and concatenated. After centering and scaling to align the variances, and SVD to reduce dimensionality, a sequence of 2000 10-dimensional processed observations was obtained.

For the experiment, 15 sequences of 200 frames were used to learn models of the environment via EM initialized by subspace ID; of these models 10 were unstable. The unstable model was stabilized using constraint generation, LB-1 and LB-2. Stabilization was performed in the last M-step of EM (doing it in every EM iteration is more expensive and did not improve performance). To test predictive power, filtering was performed for 20 frames using the original model, and from there on, prediction was performed for different extents ranging from 1 to 500, from the original unstable model and the three stabilized models. This filtering and prediction was repeated on 1500 separate subsequences.

The resulting plot of prediction loglikelihood over different prediction extents (Figure 5.4(E)) shows that the model stabilized using constraint generation has the higher prediction loglikelihood of the unstable model in the short term (1-120 timesteps), while in the long term all the stable models have a higher prediction loglikelihood than the unstable model, whose score declines steadily while the stable models' score remains higher consistently.

5.4.3 Stable Baseline Models for Biosurveillance

We examine daily counts of OTC drug sales in pharmacies, obtained from the National Data Retail Monitor (NDRM) collection [67]. The counts are divided into 23 different categories and are tracked separately for each zipcode in the country. We focus on zipcodes from a particular American city. The data exhibits 7-day periodicity due to differential buying patterns during weekdays and weekends. We isolate a 60-day subsequence where the data dynamics remain relatively stationary, and attempt to learn LDS parameters to be able to simulate sequences of baseline values for use in detecting anomalies. In principle, more accurate baseline models should lead to better anomaly detection.

We perform two experiments on different aggregations of the OTC data, with parameter values $n = 7, d = 7$ and $\tau = 14$. Figure 5.5(A) plots 22 different drug categories aggregated over all zipcodes, and Figure 5.5(B) plots a single drug category (cough/cold) in 29 different zipcodes separately. In both cases, constraint generation is able to use very little training data to learn a stable model that captures the periodicity in the data, while the least squares model is unstable and its predictions diverge over time. LB-1 learns a model that is stable but overconstrained, and the simulated observations quickly drift from the correct magnitudes.

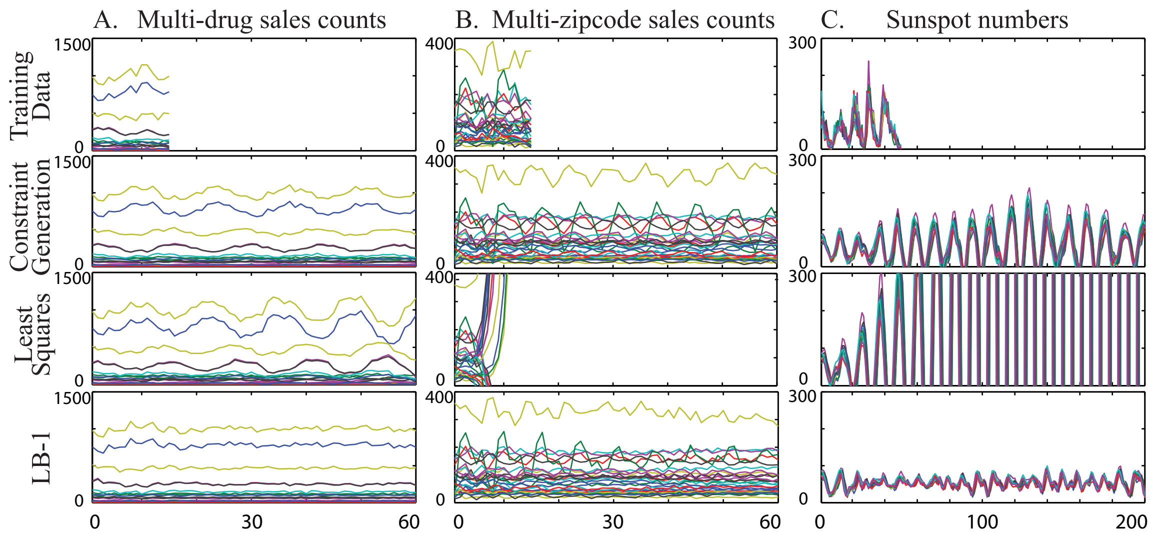


Figure 5.5: (A): 60 days of data for 22 drug categories aggregated over all zipcodes in the city. (B): 60 days of data for a single drug category (cough/cold) for all 29 zipcodes in the city. (C): Sunspot numbers for 200 years separately for each of the 12 months. The training data (top), simulated output from constraint generation, output from the unstable least squares model, and output from the over-damped LB-1 model (bottom).

5.4.4 Modeling Sunspot Numbers

We compared least squares and constraint generation on learning LDS models for the sunspot data discussed earlier in Section 3.3.2. We use parameter settings $n = 7, d = 18, \tau = 50$. Figure 5.5(C) represents a data-poor training scenario where we train a least-squares model on 18 timesteps, yielding an unstable model whose simulated observations increase in amplitude steadily over time. Again, constraint generation is able to use very little training data to learn a stable model that seems to capture the periodicity in the data as well as the magnitude, while the least squares model is unstable. The model learned by LB-1 attenuates more noticeably, capturing the periodicity to a smaller extent. Quantitative results on both these domains exhibit similar trends as those in Table 5.1.

5.5 Discussion

We have introduced a novel method for learning stable linear dynamical systems. Our constraint generation algorithm is more powerful than previous methods in the sense of optimizing over a larger set of stable matrices with a suitable objective function. In practice, the benefits of stability guarantees are readily noticeable, especially when the training data is limited. This connection between stability and amount of training data is important in practice, since time series data is often expensive to collect in large quantities, especially for phenomena with long periods in domains like physics or astronomy. The constraint generation approach also has the benefit of being faster than previous methods in nearly all of our experiments. Stability could also be of advantage in planning applications. Subspace ID, and its stable variant introduced here, provide a different way of looking at LVM parameter learning, one that is based on optimizing the *predictive* capabilities of the model. Depending on the number of observations stacked in the Hankel matrix, we could optimize for short-term or long-term prediction if the data domain is expected or known to have short or long-range periodicity. The simplicity of the matrix decomposition approach, and its natural method for model selection by examining singular values, is also very attractive in contrast to EM-based approaches. These factors motivate us to seek an analogous learning method for HMM-type models as well, to realize these benefits in the discrete LVM domain. In the next chapter we propose a variant of HMMs, along with a matrix-decomposition-based learning algorithm for it that attains this goal.

Chapter 6

Reduced-Rank Hidden Markov Models

So far we have examined continuous-observation Hidden Markov Models (HMMs) [4] and Linear Dynamical Systems (LDSs) [28], which are two examples of latent variable models of dynamical systems. The distributional assumptions of HMMs and LDSs result in important differences in the evolution of their belief over time. The discrete state of HMMs is good for modeling systems with mutually exclusive states that can have completely different observation signatures. The predictive distribution over observations is allowed to be non-log-concave when predicting or simulating the future, leading to what we call *competitive inhibition* between states (see Figure 6.4 below for an example). Competitive inhibition denotes the ability of a model's predictive distribution to place probability mass on observations while disallowing mixtures of those observations. Conversely, the Gaussian predictive distribution over observations in LDSs is log-concave, and thus does not exhibit competitive inhibition. However, LDSs naturally model *smooth state evolution*, which HMMs are particularly bad at. The dichotomy between the two models hinders our ability to compactly model systems that exhibit *both* competitive inhibition and smooth state evolution. In this chapter we present the *Reduced-Rank Hidden Markov Model (RR-HMM)*, a dynamical system model which can perform smooth state evolution as well as competitive inhibition. Intuitively the RR-HMM assumes that the dynamical system evolves smoothly along a low-dimensional subspace in a large discrete state space. We discuss theoretical connections to previous work, propose a learning algorithm with finite-sample performance guarantees, and demonstrate results on high-dimensional real-world sequential data.

6.1 Introduction

HMMs can approximate smooth state evolution by tiling the state space with a very large number of low-observation-variance discrete states with a specific transition structure. However, inference and learning in such a model is highly inefficient due to the large number of parameters, and due to the fact that existing HMM learning algorithms, such as Expectation Maximization (EM) [4], are prone to local minima. More sophisticated EM-based algorithms for learning HMMs that avoid local minima, such as STACS (Chapter 4), help to some extent. However they are still heuristics that do not offer any theoretical performance guarantees, and cannot learn very large-state-space models as efficiently or accurately as we would like (on the other hand, STACS does learn a *positive realization* of HMM parameters unlike the spectral method we describe here, but this does not hinder us from performing inference and prediction using the RR-HMM parameters we obtain). RR-HMMs allow us to reap many of the benefits of large-state-space HMMs without incurring the associated inefficiency during inference and learning. Indeed, we show that all inference operations in the RR-HMM can be carried out in the low-dimensional space where the dynamics evolve, decoupling their computational cost from the number of hidden states. This makes *rank-k* RR-HMMs (with any number of states) as computationally efficient as *k-state* HMMs, but much more expressive. Figure 6.1(A) shows an example of observations from a 4-state dynamical system which cannot be represented by a 3-state HMM Figure 6.1(B), but is accurately modeled by a rank-3 RR-HMM (Figure 6.1(C)) that can be learned using a single invocation of the algorithm described here. A 4-state HMM can be learned for this data (Figure 6.1(D)). Traditional methods for learning this HMM, such as EM, involves performing multiple restarts to escape local minima.

Though the RR-HMM is in itself novel, its low-dimensional $\mathbb{R}^k$ representation is a special case of several existing models such as Predictive State Representations [68], Observable Operator Models [69], generalized HMMs [70] and multiplicity automata [71, 72],

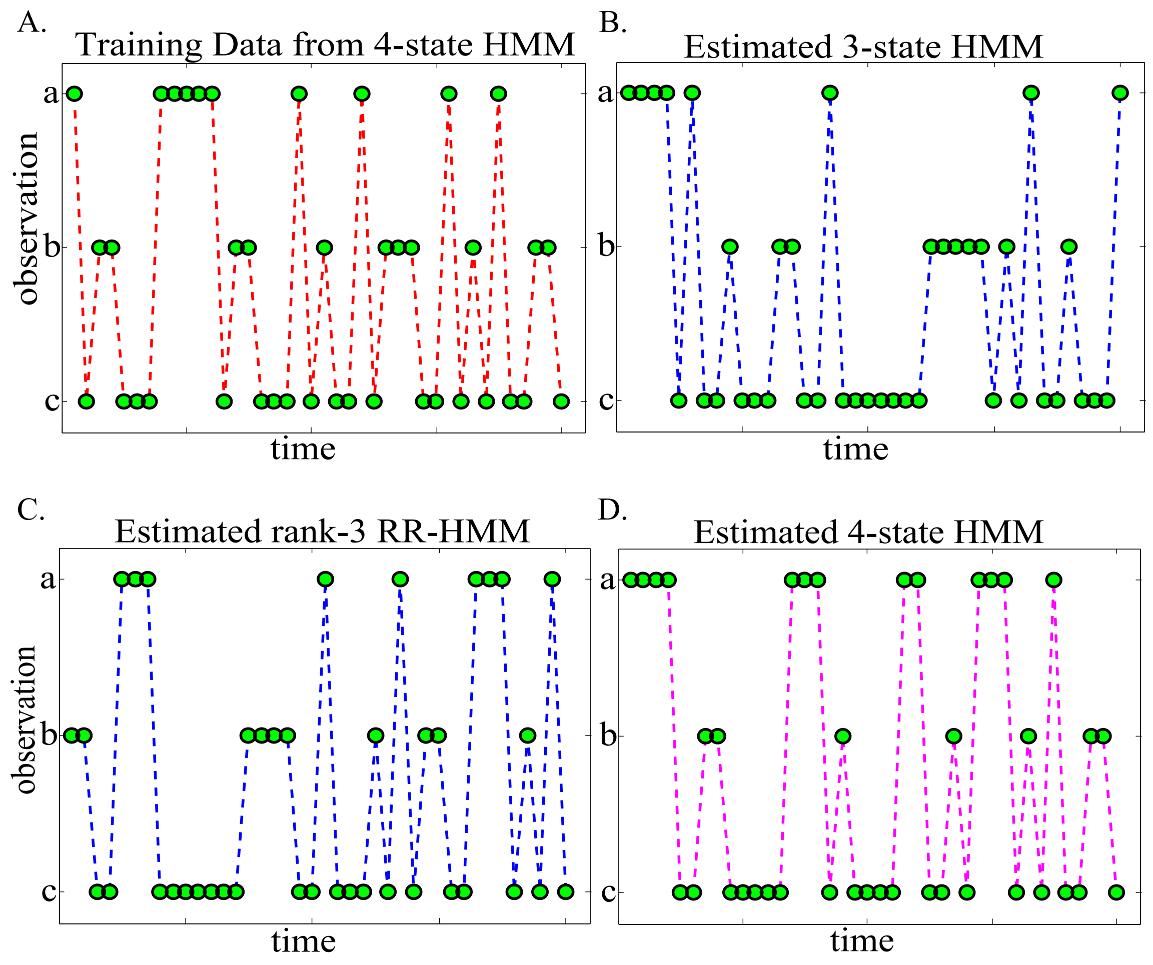


Figure 6.1: (A) Observations from a dynamical system with 4 discrete states and 3 discrete observations, two of whose states emit the observation ‘c’ and can only be distinguished by the underlying dynamics. (B) 3-state HMMs learned using EM with multiple restarts cannot represent this model, as evinced by simulations from this model. (C) A rank-3 RR-HMM estimated using a single run of the learning algorithm described in this chapter represents this 4-state model accurately (as seen from simulations), as does (D) a 4-state HMM learned using EM, though the latter needs multiple restarts to discover the overlapping states and avoid local minima.

and is also related to the representation of LDSs learned using Subspace Identification [27]. These and other related models and algorithms are discussed further in Section 6.4.

To learn RR-HMMs from data, we adapt and extend a recently proposed spectral learning algorithm by Hsu, Kakade and Zhang [12] (henceforth referred to as HKZ) that learns *observable representations* of HMMs using matrix decomposition and regression on empirically estimated observation probability matrices of past and future observations. An observable representation of an HMM allows us to model sequences with a series of operators without knowing the underlying stochastic transition and observation matrices. The HKZ algorithm is free of local optima and asymptotically unbiased, with finite-sample bounds on L_1 error in joint probability estimates and on KL-divergence of conditional probability estimates from the resulting model. However, the original algorithm and its bounds assume (1) that the transition model is full-rank and (2) that single observations are informative about the entire latent state, i.e. 1-*step observability*. We generalize the HKZ bounds to the low-rank transition matrix case and derive tighter bounds that depend on k instead of m, allowing us to learn rank-k RR-HMMs of arbitrarily large m in $\mathcal{O}(Nk^2)$ time, where N is the number of samples. We also describe and test a method for circumventing the 1-step observability condition by combining observations to make them more informative. Furthermore, in the Appendix we also provide consistency results which show that the learning algorithm in fact can learn PSRs (more details on this are available in [73], though our error bounds don't yet generalize to these models.

Experiments show that our learning algorithm can recover the underlying RR-HMM in a variety of synthetic domains. We also demonstrate that RR-HMMs are able to compactly model smooth evolution *and* competitive inhibition in a clock pendulum video, as well as in real-world mobile robot vision data captured in an office building. Robot vision data (and, in fact, most real-world multivariate time series data) exhibits smoothly evolving dynamics requiring multimodal predictive beliefs, for which RR-HMMs are particularly suited. We compare performance of RR-HMMs to LDSs and HMMs on simulation and prediction tasks. Proofs and some other details are in the Appendix.

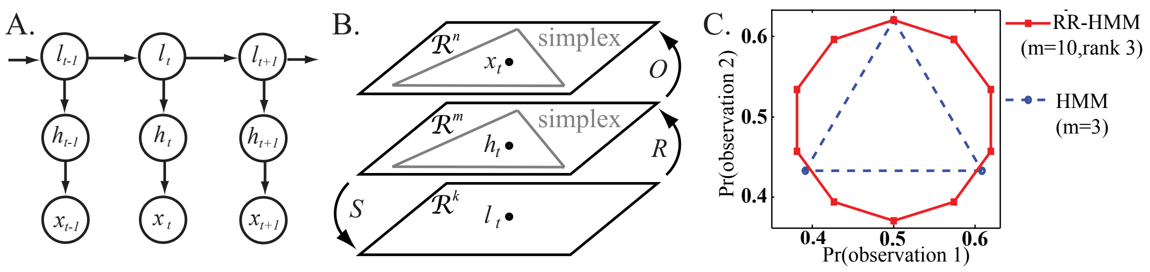


Figure 6.2: (A) The graphical model representation of an RR-HMM. l_t denotes the k-dimensional state vector, h_t the m-dimensional discrete state, and x_t the discrete observation. The distributions over h_t and l_{t+1} are deterministic functions of l_t. (B) An illustration of different RR-HMM parameters and the spaces and random variables they act on. (C) Projection of sets of predictive distributions of a rank 3 RR-HMM with 10 states, and a 3-state full-rank HMM with similar parameters.

6.1.1 Definitions

Assume the HMM notation introduced in Chapter 2, assume T has rank k and let $T = RS$ where $R \in \mathbb{R}^{m\times k}$ and $S \in \mathbb{R}^{k\times m}$. This implies that the dynamics of the system can be expressed in $\mathbb{R}^k$ rather than $\mathbb{R}^m$. By convention, we think of S as projecting the m-dimensional state distribution vector to a k-dimensional state vector, and R as expanding this low-dimensional state back to an m-dimensional state distribution vector while propagating it forward in time. One possible choice for R and S is to use any k independent columns of T as the columns of R, and let the columns of S contain the coefficients required to reconstruct T from R, though other choices are possible (e.g. using SVD). Also assume for now that $m \leq n$ (we relax this assumption in Section 6.2.4). We denote the k-dimensional projection of the hidden state vector $\vec{h}_t$ as $\vec{l}_t$, which is simply a vector of real numbers rather than a stochastic vector. We assume the initial state distribution lies in the low dimensional space as well, i.e. $\vec{\pi} = R\vec{\pi}_l$ for some vector $\vec{\pi}_l \in \mathbb{R}^k$. Figure 6.2(A) illustrates the graphical model corresponding to an RR-HMM. Figure 6.2(B) illustrates some of the different RR-HMM parameters and the spaces they act on.

To see how the probability of a sequence can be computed using these parameters, first recall the definition of $A_x \in \mathbb{R}^{m\times m}$ according to equation (2.1) and the fact that $T = RS$:

$$A_x = RS \ \text{diag}(O_{x,\cdot})$$

and define $W_x \in \mathbb{R}^{k\times k}$

$$W_x = S \ \text{diag}(O_{x,\cdot})R$$

Also let $W = \sum_x W_x = SR$. Equation (2.2) shows how to write the joint probability of a sequence $x_1, ..., x_t$ using $\{A_x\}$. With the above definitions, the joint probability can also be written using $\{W_x\}$ as well:

$$\begin{aligned}
\Pr[x_1, ..., x_t] &= \vec{1}_m^\top RS \ \text{diag}(O_{x_t,\cdot})RS \ \text{diag}(O_{x_{t-1},\cdot})R \cdots S \ \text{diag}(O_{x_1,\cdot})\vec{\pi} \quad \text{(from equation (2.2))} \\
&= \vec{1}_m^\top R \left(S \ \text{diag}(O_{x_t,\cdot})R\right) \left(S \ \text{diag}(O_{x_{t-1},\cdot})R\right) \cdots \left(S \ \text{diag}(O_{x_1,\cdot})R\right) S\vec{\pi} \\
&= \vec{1}_m^\top R W_{x_t} \ldots W_{x_1} \vec{\pi}_l \quad \text{(by definition of } W_x, \vec{\pi}_l\text{)} \qquad (6.1)
\end{aligned}$$

The latter parametrization casts a rank-k RR-HMM as a k-dimensional PSR or transformed PSR [74]. Inference can be carried out in $\mathcal{O}(Nk^2)$ time in this representation. However, since every HMM is trivially a PSR, this leads to the question of how expressive rank-k RR-HMMs are in comparison to k-state full-rank HMMs. The following example is instructive.

6.1.2 Expressivity of RR-HMMs

We describe a rank-k RR-HMM whose set of possible predictive distributions is easy to visualize and describe. Consider the following rank 3 RR-HMM with 10 states and 4 observations. The observation probabilities in each state are of the form

$$O_{i,\cdot} \ = \ [\, p_i q_i \quad p_i(1-q_i) \quad (1-p_i)q_i \quad (1-p_i)(1-q_i)]$$

for some $0 \le p_i, q_i \le 1$, which can be interpreted as 4 discrete observations, factored as two binary components which are independent given the state. T and p_i, q_i are chosen to place the vertices of the set of possible predictive distributions on evenly spaced points along a circle in (p, q)-space:

$$\begin{aligned} T_{ij} &= (1/2m)\left[2 + \sin(2\pi i/m)\sin(2\pi j/m) + \cos(2\pi i/m)\cos(2\pi j/m)\right] \\ p_i &= \left[\sin(2\pi i/m) + 1\right]/2 \\ q_i &= \left[\cos(2\pi i/m) + 1\right]/2 \end{aligned}$$

We plot the marginal probability of each component of the observation, ranging across all achievable values of the latent state vector for the $m = 10$ case (Figure 6.2(C)), yielding a 10-sided polygon as the projection of the set of possible predictive distributions. These distributions are the columns of $T^{\mathsf{T}}O$. We also plot the corresponding marginals for the $m = 3$ full-rank HMM case to yield a triangular set. More generally, from a k-state HMM, we can get at most a k-sided polygon for the set of possible predictive distributions.

The above example illustrates that rank-k RR-HMMs with m states can model sets of predictive distributions which full-rank HMMs with less than m states cannot express. However, as we shall see, inference in rank-k RR-HMMs of arbitrary m is as efficient as inference in k-state full-rank HMMs. This implies that the additional degrees of freedom in the RR-HMM's low-dimensional parameters and state vectors buy it considerable expressive power. Since RR-HMMs are also related to PSRs as pointed out in the previous section, and since our learning algorithm will be shown to be *consistent* for estimating PSRs (though we have finite-sample guarantees only for the RR-HMM case), it is also instructive to examine the expressivity of PSRs in general. We refer the reader to Jaeger (2000) [69] and James et. al. (2004) [75] for more on this.

6.2 Learning Reduced-Rank HMMs

In a full-rank HMM, the maximum likelihood solution for the parameters $\{T, O\}$ can be found through iterative techniques such as expectation maximization (EM) [15]. EM, however, is prone to local optima and does not address the model selection problem. STACS is better but still not guaranteed to return anything close to optimal as data increases, and is slow beyond a certain state space magnitude. Moreover, in learning RR-HMMs we face the additional challenge of learning the factors of its low-rank transition matrix. We could use EM to estimate T followed by (or combined with) matrix factorization algorithms such as Singular Value Decomposition (SVD) [32] or Non-negative Matrix Factorization (NMF) [76]. This approach has several drawbacks. For example, if the noisy estimate of a low-rank transition matrix is not low-rank itself, SVD could cause negative numbers to appear in the reconstructed transition matrix. Also, algorithms for NMF are only locally optimal, and NMF is overly restrictive in that it constrains its factor matrices to be non-negative, which is unnecessary for our application since low-rank transition matrices may have negative numbers in their factors R and S.

An alternative approach, which we adopt, is to learn an asymptotically unbiased observable representation of an RR-HMM directly using SVD of a probability matrix relating past and future observations. This idea has roots in subspace identification [27, 29] and multiplicity automata [71, 72, 70] as well as the PSR/OOM literature [69, 77] and was recently formulated in a paper by Hsu, Kakade and Zhang [12] for full-rank HMMs. We use their algorithm, extending its theoretical guarantees for the low-rank HMM case where the rank of the transition matrix T is $k \leq m$. Computationally, the only difference in our base algorithm (before Section 6.2.4) is that we learn a rank k representation instead of rank m. This allows us learn much more compact representations of possibly large-state-space real-world HMMs, and greatly increases the applicability of the original algorithm. Even when the underlying HMM is not low-rank, we can examine the singular values to tune the complexity of the underlying RR-HMM, thus providing a natural method for model selection. We present the main definitions, the algorithm and its performance bounds below. Detailed versions of the supporting proofs and lemmas can be found in the Appendix.

6.2.1 The Algorithm

The learning algorithm depends on the following vector and matrix quantities that comprise properties of single observations, pairs of observations and triples:

$$
\begin{aligned}
[P_1]_i &= \Pr[x_1 = i] \\
[P_{2,1}]_{i,j} &= \Pr[x_2 = i, x_1 = j] \\
[P_{3,x,1}] &= \Pr[x_3 = i, x_2 = x, x_1 = j] \text{ for } x = 1, \ldots, n
\end{aligned}
$$

$P_1 \in \mathbb{R}^n$ is a vector, $P_{2,1} \in \mathbb{R}^{n \times n}$ and $P_{3,x,1} \in \mathbb{R}^{n \times n}$ are matrices. These quantities are closely related to matrices computed in algorithms for learning OOMs [69], PSRs [77] and LDSs using subspace identification (Subspace ID) [27]. They can be expressed in terms of HMM parameters (for proofs see the Appendix: Lemmas 8 and 9 in Section A.1.1):

$$
\begin{aligned}
\vec{P}_1^\top &= \vec{1}_m^\top T \ \operatorname{diag}(\pi) O^\top \\
P_{2,1} &= OT \ \operatorname{diag}(\pi) O^\top \\
P_{3,x,1} &= OA_x T \ \operatorname{diag}(\pi) O^\top
\end{aligned}
$$

Note that $P_{2,1}$ and $P_{3,x,1}$ both contain a factor of T and hence are both of rank k for a rank-k RR-HMM. This property will be important for recovering an estimate of the RR-HMM parameters from these matrices. The primary intuition is that, when projected onto an appropriate linear subspace, $P_{3,x,1}$ is linearly related to $P_{2,1}$ through a product of RR-HMM parameters. This allows us to devise an algorithm that

1. estimates $P_{2,1}$ and $P_{3,x,1}$ from data,
2. projects them to an appropriate linear subspace computed using SVD,
3. uses linear regression to estimate the RR-HMM parameters (up to a similarity transform) from these projections.

Specifically, the algorithm attempts to learn an *observable representation* of the RR-HMM using a matrix $U \in \mathbb{R}^{n \times k}$ such that $U^\top OR$ is invertible. An observable representation is defined as follows.

Definition 1 *The* <u>*observable representation*</u> *is defined to be the parameters* $b_1, b_\infty, \{B_x\}_{x=1}^n$ *such that:*

$$\vec{b}_1 = U^\top P_1 \tag{6.2a}$$

$$\vec{b}_\infty = (P_{2,1}^\top U)^+ P_1 \tag{6.2b}$$

$$B_x = (U^\top P_{3,x,1})(U^\top P_{2,1})^+ \quad \textit{for } x = 1, \ldots, n \tag{6.2c}$$

For the RR-HMM, note that the dimensionality of the parameters is determined by k, not m: $b_1 \in \mathbb{R}^k$, $b_\infty \in \mathbb{R}^k$ and $\forall x \quad B_x \in \mathbb{R}^{k \times k}$. Though these definitions seem arbitrary at first sight, the observable representation of the RR-HMM is closely related to the true parameters of the RR-HMM in the following manner (see Lemma 9 in the Appendix for the proof):

1. $\vec{b}_1 = (U^\top OR)\pi_l = (U^\top O)\pi$,

2. $\vec{b}_\infty^\top = 1_m^\top R(U^\top OR)^{-1}$,

3. For all $x = 1, \ldots, n$: $B_x = (U^\top OR) W_x (U^\top OR)^{-1}$

Hence B_x is a similarity transform of the RR-HMM parameter matrix $W_x = S \ \mathrm{diag}(O_{x,\cdot})R$ (which, as we saw earlier, allows us to perform RR-HMM inference), and $\vec{b}_1$ and $\vec{b}_\infty$ are the corresponding linear transformations of the RR-HMM initial state distribution and the RR-HMM normalization vector. Note that $(U^\top OR)$ must be invertible for these relationships to hold. Together, the parameters $\vec{b}_1$,$\vec{b}_\infty$ and B_x for all x comprise the observable representation of the RR-HMM. Our learning algorithm will estimate these parameters from data. The algorithm for estimating rank-k RR-HMMs is equivalent to the spectral HMM learning algorithm of HKZ [12] for learning k-state HMMs. Our relaxation of their conditions (e.g. HKZ assume a full-rank transition matrix, without which their bounds are vacuous), and our performance guarantees for learning rank-k RR-HMMs, show that the algorithm learns a much larger class of k-dimensional models than the class of k-state HMMs.

LEARN-RR-HMM(k, N) The learning algorithm takes as input the desired rank k of the underlying RR-HMM rather than the number of states m. Alternatively, given a singular value threshold the algorithm can choose the rank of the HMM by examining the singular values of $P_{2,1}$ in Step 2. It assumes that we are given N independently sampled observation triples (x_1, x_2, x_3) from the HMM. In practice, we can use a single long sequence of observations as long as we discount the bound on the number of samples based on the mixing rate of the HMM (i.e. (1 $-$ the second eigenvalue of T)), in which case π must correspond to the stationary distribution of the HMM to allow estimation of $\vec{P}_1$. The algorithm results in an *estimated observable representation* of the RR-HMM, with parameters $\widehat{b}_1, \widehat{b}_\infty$, and $\widehat{B}_x$ for $x = 1, \ldots, n$. The steps are briefly summarized here for reference:

1. Compute empirical estimates $\widehat{P}_1, \widehat{P}_{2,1}, \widehat{P}_{3,x,1}$ of $\vec{P}_1, P_{2,1}, P_{3,x,1}$ (for $x = 1, ..., n$).

2. Use SVD on $\widehat{P}_{2,1}$ to compute $\widehat{U}$, the matrix of left singular vectors corresponding to the k largest singular values.

3. Compute model parameter estimates:

   (a) $\widehat{b}_1 = \widehat{U}^\top \widehat{P}_1$,

   (b) $\widehat{b}_\infty = (\widehat{P}_{2,1}^\top \widehat{U})^+ \widehat{P}_1$,

   (c) $\widehat{B}_x = \widehat{U}^\top \widehat{P}_{3,x,1} (\widehat{U}^\top \widehat{P}_{2,1})^+$ (for $x = 1, \ldots, n$)

We now examine how we can perform inference in the RR-HMM using the observable representation. For this, we will need to define the *internal state* $\vec{b}_t$. Just as the parameter $\vec{b}_1$ is a linear transform of the initial RR-HMM belief state, $\vec{b}_t$ is a linear transform of the belief state of the RR-HMM at time t (Lemma 10 in Section A.1.1 of the Appendix):

$$\vec{b}_t = (U^\top O R) \vec{l}_t(x_{1:t-1}) = (U^\top O) \vec{h}_t(x_{1:t-1})$$

This internal state $\vec{b}_t$ can be updated to condition on observations and evolve over time, just as we can update $\vec{l}_t$ for RR-HMMs and $\vec{h}_t$ for regular HMMs.

6.2.2 Inference in the Observable Representation

Given a set of observable parameters, we can predict the probability of a sequence, update the internal state $\widehat{b}_t$ to perform filtering and predict conditional probabilities as follows (see Lemma 10 in the Appendix for proof):

- Predict sequence probability: $\widehat{\Pr}[x_1, \ldots, x_t] = \widehat{b}_\infty^\mathsf{T} \widehat{B}_{x_t} \ldots \widehat{B}_{x_1} \widehat{b}_1$
- Internal state update: $\widehat{b}_{t+1} = \frac{\widehat{B}_{x_t}\widehat{b}_t}{\widehat{b}_\infty^\mathsf{T} \widehat{B}_{x_t}\widehat{b}_t}$
- Conditional probability of x_t given $x_{1:t-1}$: $\widehat{\Pr}[x_t \mid x_{1:t-1}] = \frac{\widehat{b}_\infty^\mathsf{T} \widehat{B}_{x_t}\widehat{b}_t}{\sum_x \widehat{b}_\infty^\mathsf{T} \widehat{B}_x \widehat{b}_t}$

Estimated parameters can, in theory, lead to negative probability estimates. These are most harmful when they cause the normalizers $\widehat{b}_\infty^\mathsf{T} \widehat{B}_{x_t}\widehat{b}_t$ or $\sum_x \widehat{b}_\infty^\mathsf{T} \widehat{B}_x \widehat{b}_t$ to be negative. However, in our experiments, the latter was never negative and the former was very rarely negative; and, using real-valued observations with KDE (as in Section 6.2.6) makes negative normalizers even less likely, since in this case the normalizer is a weighted sum of several estimated probabilities. In practice we recommend thresholding the normalizers with a small positive number, and not trusting probability estimates for a few steps if the normalizers fall below the threshold.

Note that the inference operations occur entirely in $\mathbb{R}^k$. We mentioned earlier that parameterizing RR-HMM parameters as W_x for all observations x casts it as a PSR of k dimensions. In fact the learning and inference algorithms for RR-HMMs proposed here have no dependence on the number of states m whatsoever, though other learning algorithms for RR-HMMs can depend on m (e.g. if they learn R and S directly). The RR-HMM formulation is intuitively appealing due to the idea of a large discrete state space with low-rank transitions, but this approach is also a provably *consistent learning algorithm for PSRs* in general, with finite-sample performance guarantees for the case where the PSR is an RR-HMM. Since PSRs are provably more expressive and compact than finite-state HMMs [69, 75], this indicates that we can learn a more powerful class of models than HMMs using this algorithm.

6.2.3 Theoretical Guarantees

The following finite sample bound on the estimated model generalizes analogous results from HKZ to the case of low-rank T. Theorem 2 bounds the L_1 error in joint probability estimates from the learned model. This bound shows the consistency of the algorithm in learning a correct observable representation of the underlying RR-HMM, without ever needing to recover the high-dimensional parameters R, S, O of the latent representation. Note that our error bounds are *independent* of m, the number of hidden states; instead, they depend on k, the rank of the transition matrix, which can be much smaller than m. Since HKZ explicitly assumes a full-rank HMM transition matrix, and their bounds become vacuous otherwise, generalizing their framework involves relaxing this condition, generalizing the theoretical guarantees of HKZ and deriving proofs for these guarantees.

Define $\sigma_k(M)$ to denote the k^{th} largest singular value of matrix M. The sample complexity bounds depend polynomially on $1/\sigma_k(P_{2,1})$ and $1/\sigma_k(OR)$. The larger $\sigma_k(P_{2,1})$ is, the more well-separated are the dynamics from noise. The larger $\sigma_k(OR)$ is, the more informative the observation is regarding state. For both these quantities, the larger the magnitude, the fewer samples we need to learn a good model. The bounds also depend on a term $n_0(\epsilon)$, which is the minimum number of observations that account for $(1 - \epsilon)$ of the total probability mass, i.e. the number of "important" observations. Recall that N is the number of independently sampled observation triples which comprise the training data, though as mentioned earlier we can also learn from a single long training sequence.

The theorem holds under mild conditions. Some of these are the same as (or relaxations of) conditions in HKZ, namely that the prior $\vec{\pi}$ is nonzero everywhere, and a number of matrices of interest $(R, S, O, (U^\top OR))$ are of rank at least k for invertibility reasons. The other conditions are unique to the low-rank setting, namely that $S \ \mathrm{diag}(\vec{\pi})O^\top$ has rank at least k, R has at least one column whose L_2 norm is at most $\sqrt{k/m}$, and the L_1 norm of R is at most 1. The first of these conditions implies that the column space of S and the row space of O have some degree of overlap. The other two are satisfied, in the case of HMMs, by thinking of R as containing k linearly independent probability distributions along its columns (including a near-uniform column) and of S as containing the coefficients needed to obtain T from those columns. Alternatively, the conditions can be satisfied for an arbitrary R by scaling down entries of R and scaling up entries of S accordingly. However, this increases $1/\sigma_k(OR)$, and hence we pay a price by increasing the number of samples needed to attain a particular error bound. See the Appendix (Section A.1.1) for formal statements of these conditions.

Theorem 2 *[Generalization of HKZ Theorem 6] There exists a constant $C > 0$ such that the following holds. Pick any $0 \leq \epsilon, \eta \leq 1$ and $t \geq 1$. Assume the HMM obeys Conditions 3,4,5,6 and 7. Let $\varepsilon = \sigma_k(OR)\sigma_k(P_{2,1})\epsilon/(4t\sqrt{k})$. Assume*

$$N \geq C \cdot \frac{t^2}{\epsilon^2} \cdot \left(\frac{k}{\sigma_k(OR)^2 \sigma_k(P_{2,1})^4} + \frac{k \cdot n_0(\varepsilon)}{\sigma_k(OR)^2 \sigma_k(P_{2,1})^2} \right) \cdot \log(1/\eta)$$

With probability $\geq 1 - \eta$, the model returned by LEARNRR-HMM*(k, N) satisfies*

$$\sum_{x_1,\ldots,x_t} | \Pr[x_1, \ldots, x_t] - \widehat{\Pr}[x_1, \ldots, x_t]| \leq \epsilon$$

where the summation is over all possible hidden state sequences of length t.

For the proof, see the Appendix (Section A.1.4).

6.2.4 Learning with Observation Sequences as Features

The probability matrix $P_{2,1}$ relates one past timestep to one future timestep, under the assumption that the vector of observation probabilities at a single step is sufficient to disambiguate state ($n \geq m$ and $\text{rank}(O) = m$). In system identification theory, this corresponds to assuming 1-*step observability* [27]. This assumption is unduly restrictive for many real-world dynamical systems of interest. More complex sufficient statistics of past and future may need to be modeled, such as the *block Hankel matrix* formulations for subspace methods [27, 29] to identify linear systems that are not 1-step observable.

For RR-HMMs, this corresponds to the case where $n < m$ and/or $\text{rank}(O) < m$. Similar to the Hankel matrix formulation, we can stack multiple observation vectors such that each augmented observation comprises data from several, possibly consecutive, timesteps. The observations in the augmented observation vectors are assumed to be non-overlapping, i.e. all observations in the new observation vector at time $t+1$ have larger time indices than observations in the new observation vector at time t. This corresponds to assuming *past sequences* and *future sequences* spanning multiple timesteps as events that characterize the dynamical system, causing $\vec{P}_1$,$P_{2,1}$ and $P_{3,x,1}$ to be larger. Note that the x in $P_{3,x,1}$ still denotes a *single* observation, whereas the other indices in $\vec{P}_1$, $P_{2,1}$ and $P_{3,x,1}$ are now associated with events. For example, if we stack $\overline{n}$ consecutive observations, $P_{3,x,1}[i,j]$ equals the probability of seeing the i^{th} $\overline{n}$-length sequence, followed by the single observation x, followed by the j^{th} $\overline{n}$-length sequence. Empirically estimating this matrix consists of scanning for the appropriate subsequences i and j separated by observation symbol x, and normalizing to obtain the occurrence probability.

$P_{2,1}$ and $P_{3,x,1}$ become larger matrices if we use a larger set of events in the past and future. However, stacking observations does not complicate the *dynamics*: it can be shown that the rank of $P_{2,1}$ and $P_{3,x,1}$ cannot exceed k (see Section A.2 in the Appendix for a proof sketch). Since our learning algorithm relies on an SVD of $P_{2,1}$, this means that augmenting the observations does not increase the rank of the HMM we are trying to recover. Also, since $P_{3,x,1}$ is still an observation probability matrix with respect to a *single* unstacked observation x in the middle, the number of observable operators we need remains constant. Our complexity bounds successfully generalize to this case, since they only rely on $\vec{P}_1$, $P_{2,1}$ and $P_{3,x,1}$ being matrices of probabilities summing to 1 (for the former two) or to $\Pr[x_2 = x]$ (for the latter), as they are here.

The extension given above for learning HMMs with ambiguous observations differs from the approach suggested by HKZ, which simply substitutes observations with *overlapping* tuples of observations (e.g. $\overline{P_{2,1}}(j,i) = \Pr[x_3 = j_2, x_2 = j_1, x_2 = i_2, x_1 = i_1]$). There are two potential problems with the HKZ approach. First, the number of observable operators increases exponentially with the length of each tuple: there is one observable operator per tuple, instead of one per observation. Second, $\overline{P_{2,1}}$ cannot be decomposed into a product of matrices that includes T, and consequently no longer has rank equal to the rank of the HMM being modeled. Thus, the learning algorithm could require much more data to recover a correct model if we use the HKZ approach.

6.2.5 Learning with Indicative and Characteristic Features

To generalize even further beyond sequences of observations, we can think about deriving higher-level features of past and future observations, which we call *indicative features* and *characteristic features* respectively, and generalizing $\vec{P}_1, P_{2,1} P_{3,x,1}$ to contain *expected values* of these features (or possibly products of features). These are analogous to but more general than the *indicative events* and *characteristic events* of OOMs [69] and *suffix-histories* and *tests* of PSRs [77]. These more specific formulations can be obtained by assuming our indicative and characteristic features to be *indicator variables* of some indicative and characteristic events, causing the expected values of singleton, pairs and triples of these features (or their products) to equal probabilities of occurrence of these events. This leads to the current scenario where the matrices $\vec{P}_1, P_{2,1}$ and $P_{3,x,1}$ contain probabilities, and in fact our theoretical guarantees hold in this case. In the more general case where we choose arbitrary indicative and characteristic features of past and future observations which do not lead to a probabilistic interpretation of $\vec{P}_1, P_{2,1}$ and $P_{3,x,1}$, our error bounds do not hold as written, although there is good reason to believe that they can be generalized. However the algorithm is still valid in the sense that we can prove *consistency* of the resulting estimates. See Section A.4 of Appendix A.

6.2.6 Kernel Density Estimation for Continuous Observations

The default RR-HMM formulation assumes discrete observations. However, since the model formulation converts the discrete observations into m-dimensional probability vectors, and the filtering, smoothing and learning algorithms we discuss all do the same, it is straightforward to model multivariate continuous data. Counting on our ability to perform efficient inference and learning even in very large state spaces, we pick n points in observation space as kernels, and perform Kernel Density Estimation (KDE) to convert each continuous datapoint into a probability vector. In our experiments we use Gaussian kernels with uniform bandwidths, however any smooth kernel will suffice. The kernel centers can be chosen by sampling, clustering or other standard methods. In our experiments below, we use an initial subsequence of training data points as kernel centers directly. The kernel bandwidths can be estimated from data using maximum likelihood estimation, or can be fixed beforehand. In the limit of infinite training data, the bandwidth can shrink to zero and the KDE gives an accurate estimate of the observation density. Our theoretical results carry over to the KDE case with modifications described in Section A.1.5. Essentially, the bounds still hold for the case where we are observing stochastic vectors, though we do not yet have additional bounds connecting this existing bound to the error in estimating probabilities of raw continuous observations. When filtering in this formulation, we compute a vector of kernel center probabilities for each observation. We normalize this vector and use it to generate a convex combination of observable operators, which is used for incorporating the observation rather than any single observable operator.

This affects the learning algorithm and inference procedure as follows. Assume for ease of notation that the training data consists of N sets of three consecutive continuous observation vectors each, i.e. $\{\langle \vec{x}_{1,1}, \vec{x}_{1,2}, \vec{x}_{1,3}\rangle, \langle \vec{x}_{2,1}, \vec{x}_{2,2}, \vec{x}_{2,3}\rangle, \ldots, \langle \vec{x}_{N,1}, \vec{x}_{N,2}, \vec{x}_{N,3}\rangle\}$, though in practice we could be learning from a single long training sequence (or several). Also assume for now that each observation vector contains a single raw observation, though this technique can easily be combined with the more sophisticated sequence-based learning and feature-based learning methods described above. We will assume n kernel centers chosen from the training data, where the i^{th} kernel is centered at $\vec{c}_i$, and each kernel has a covariance matrix of Σ. λ is a bandwidth parameter that goes to zero over time. Let $\mathcal{N}(\mu, C)$ denote a multivariate Gaussian distribution with mean $\vec{\mu}$ and covariance matrix C, and $\Pr[y \mid \mathcal{N}(\mu, C)]$ be the probability of y under this Gaussian.

The learning algorithm begins by computing n-dimensional feature vectors $\langle \vec{\phi}_j \rangle_{j=1}^N$, $\langle \vec{\psi}_j \rangle_{j=1}^N$ and $\langle \vec{\xi}_j \rangle_{j=1}^N$ where

$$\begin{aligned}
[\vec{\phi}_j]_i &= \Pr[\vec{x}_{j,1} \mid \mathcal{N}(\vec{c}_i, \Sigma)] \\
[\vec{\psi}_j]_i &= \Pr[\vec{x}_{j,2} \mid \mathcal{N}(\vec{c}_i, \Sigma)] \\
[\vec{\xi}_j]_i &= \Pr[\vec{x}_{j,3} \mid \mathcal{N}(\vec{c}_i, \Sigma)]
\end{aligned}$$

In addition, the n-dimensional vectors $\langle\vec{\zeta}_j\rangle_{j=1}^N$ represent the kernel probabilities of the second observation which shrink to zero in the limit of infinite data at an appropriate rate via the bandwidth parameter λ:

$$[\vec{\zeta}_j]_i = \Pr[\vec{x}_{j,2} \mid \mathcal{N}(\vec{c}_i, \lambda\Sigma)]$$

These vectors are then normalized to sum to 1 individually. Then, the vector $\vec{P}_1$ and matrices $P_{2,1}$ and $P_{3,x,1}$ (for a given $\vec{x}$), can be estimated as follows:

$$\widehat{P}_1 = \frac{1}{N}\sum_{j=1}^{N}\vec{\phi}_j$$

$$\widehat{P}_{2,1} = \frac{1}{N}\sum_{j=1}^{N}\vec{\psi}_j\vec{\phi}_j^{\mathsf{T}}$$

$$\text{For } x = c_1, \ldots, c_n\text{: } \widehat{P}_{3,x,1} = \frac{1}{N}\sum_{j=1}^{N}[\vec{\zeta}_j]_x\vec{\xi}_j\vec{\phi}_j^{\mathsf{T}}$$

Once these matrices have been estimated, the rest of the algorithm is unchanged. We compute n 'base' observable operators $B_{c_1}, \ldots, B_{c_n}$ from the $\widehat{P}_{3,x,1}$ matrices estimated above, and vectors $\vec{b}_1$ and $\vec{b}_\infty$, as before. Given these parameter estimates, filtering for a τ-length observation sequence $\langle\vec{x}_1, \ldots, \vec{x}_\tau\rangle$ now proceeds in the following way:

```
For $t = 1, \ldots, \tau$:
    Compute $\vec{\sigma}_t$ such that $[\vec{\sigma}_t]_i = \Pr[\vec{x}_t \mid \mathcal{N}(\vec{c}_i, \lambda\Sigma)]$, and normalize.
    $B_{\sigma_t} = \sum_{j=1}^{n}[\vec{\sigma}_t]_j B_{c_j}$
    $\vec{b}_{t+1} = \frac{B_{\sigma_t}\vec{b}_t}{\vec{b}_\infty B_{\sigma_t}\vec{b}_t}$
```

6.3 Experimental Results

We designed several experiments to evaluate the properties of RR-HMMs and the learning algorithm both on synthetic and on real-world data. The first set of experiments (Section

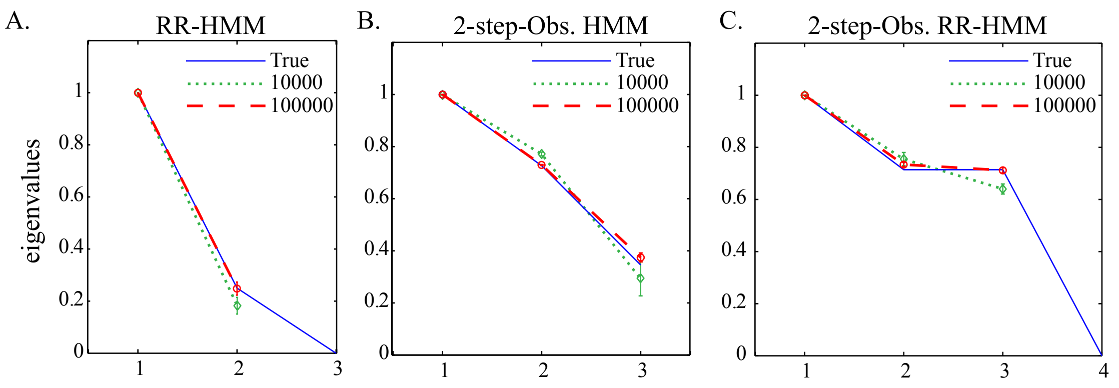


Figure 6.3: Learning discrete RR-HMMs. The three figures depict the actual eigenvalues of three different RR-HMM transition matrices, and the eigenvalues (95% error bars) of the sum of RR-HMM observable operators estimated with $10,000$ and $100,000$ training observations. (A) A 3-state, 3-observation, rank 2 RR-HMM. (B) A full-rank, 3-state, 2-observation HMM. (C) A 4-state, 2-observation, rank 3 RR-HMM.

6.3.1) tests the ability of the spectral learning algorithm to recover the correct RR-HMM. The second experiment (Section 6.3.2) evaluates the representational capacity of the RR-HMM by learning a model of a video that requires both competitive inhibition and smooth state evolution. The third set of experiments (Section 6.3.3) tests the model's ability to learn, filter, predict, and simulate video captured from a robot moving in an indoor office environment.

6.3.1 Learning Synthetic RR-HMMs

First we evaluate the unbiasedness of the spectral learning algorithm for RR-HMMs on 3 synthetic examples. In each case, we build an RR-HMM, sample observations from the model, and estimate the model with the spectral learning algorithm described in Section 6.2. We compare the eigenvalues of $B = \sum_x B_x$ in the learned model to the eigenvalues of the transition matrix T of the true model. B is a similarity transform of $S \cdot R$ which therefore has the same non-zero eigenvalues as $T = RS$, so we expect the estimated eigenvalues to converge to the true eigenvalues with enough data. This is a necessary condition for unbiasedness but not a sufficient one. See Section A.3 in Appendix for parameters of HMMs used in the examples below.

Example 1: An RR-HMM We examine an HMM with $m = 3$ hidden states, $n = 3$ observations, a full-rank observation matrix and a $k = 2$ rank transition matrix. Figure 6.3(A) plots the true and estimated eigenvalues for increasing size of dataset, along with error bars, suggesting that we recover the true dynamic model.

Example 2: A 2-step-Observable HMM We examine an HMM with $m = 3$ hidden states, $n = 2$ observations, and a full-rank transition matrix (see Appendix for parameters). This HMM violates the $m \leq n$ condition. The parameters of this HMM cannot be estimated with the original learning algorithm, since a single observation does not provide enough information to disambiguate state. By stacking 2 consecutive observations (see Section 6.2.4), however, the spectral learning algorithm can be applied successfully (Figure 6.3(B)).

Example 3: A 2-step-Observable RR-HMM We examine an HMM with $m = 4$ hidden states, $n = 2$ observations, and a $k = 3$ rank transition matrix (see Appendix for parameters). In this example, the HMM is low rank *and* multiple observations are required to disambiguate state. Again, stacking two consecutive observations in conjunction with the spectral learning algorithm is enough to recover good RR-HMM parameter estimates (Figure 6.3(C)).

6.3.2 Competitive Inhibition and Smooth State Evolution in Video

We model a clock pendulum video consisting of 55 frames (with a period of $\sim$ 22 frames) as a 10-state HMM, a 10-dimensional LDS, and a rank 10 RR-HMM with 4 stacked observations. Note that we could easily learn models with more than 10 latent states/dimensions; we limited the dimensionality in order to demonstrate the relative expressive power of the different models. For the HMM, we convert the continuous data to discrete observations

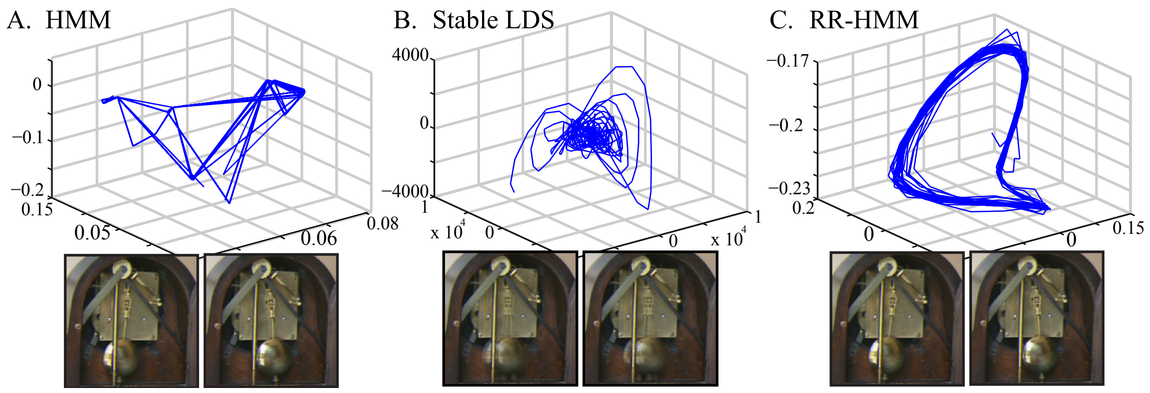


Figure 6.4: The clock video texture simulated by a HMM, a stable LDS, and a RR-HMM. (A) The clock modeled by a 10-state HMM. The manifold consists of the top 3 principal components of predicted observations during simulation. The generated frames are coherent but motion in the video is jerky. (B) The clock modeled by a 10-dimensional LDS. The manifold indicates the trajectory of the model in state space during simulation. Motion in the video is smooth but frames degenerate to superpositions. (C) The clock modeled by a rank 10 RR-HMM. The manifold consists of the trajectory of the model in the low dimensional subspace of the state space during simulation. Both the motion and the frames are correct.

by 1-NN on 25 kernel centers sampled sequentially from the training data. We trained the resulting discrete HMM using EM. We learned the LDS directly from the video using subspace ID with stability constraints [78] using a Hankel matrix of 10 stacked observations. We trained the RR-HMM by stacking 4 observations, choosing an approximate rank of 10 dimensions, and learning 25 observable operators corresponding to 25 Gaussian kernel centers. We simulate a series of 500 observations from the model and compare the manifolds underlying the simulated observations and frames from the simulated videos (Figure 6.4). The small number of states in the HMM is not sufficient to capture the smooth evolution of the clock: the simulated video is characterized by realistic looking frames, but exhibits jerky irregular motion. For the LDS, although the 10-dimensional subspace captures smooth evolution of the simulated video, the system quickly degenerates and individual frames of video are modeled poorly (resulting in superpositions of pendulums

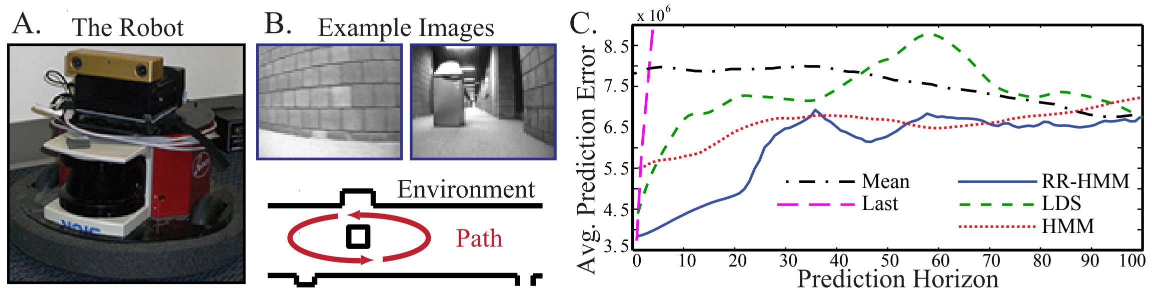


Figure 6.5: (A) The mobile robotic platform used in experiments. (B) Sample images from the robot's camera. The lower figure depicts the hallway environment with a central obstacle (black) and the path that the robot took through the environment while collecting data (the red counter-clockwise ellipse) (C) Squared loss prediction error for different models after filtering over initial part of data. The RR-HMM performs more accurate predictions consistently for 30 timesteps.

in generated frames). For the RR-HMM, the simulated video benefits from both smooth state evolution *and* competitive inhibition. The manifold in the 10-dimensional subspace is smooth and structured and the video is realistic. The results demonstrate that the RR-HMM has the benefits of smooth state evolution and compact state space of a LDS and the benefit of competitive inhibition of a HMM.

6.3.3 Filtering, Prediction, and Simulation with Robot Vision Data

We compare HMMs, LDSs, and RR-HMMs on the problem of modeling video data from a mobile robot in an indoor environment. A video of 2000 frames was collected at 6 Hz from a Point Grey Bumblebee2 stereo camera mounted on a Botrics Obot d100 mobile robot platform (Figure 6.5(A)) circling a stationary obstacle (Figure 6.5(B)) and 1500 frames were used as training data for each model. Each frame from the training data was reduced to 100 dimensions via SVD on single observations. Using this training data, we trained an RR-HMM ($k = 50, n = 1500$) using spectral learning with 20 stacked continuous observations and KDE (Section 6.2.6) with 1500 centers, a 50-dimensional LDS using

Subspace ID with Hankel matrices of 20 timesteps, and a 50-state HMM with 1500 discrete observations using EM run until convergence. For each model, we performed filtering for different extents $t_1 = 100, 101, \ldots, 250$, then predicted an image which was a further t_2 points in the future, for $t_2 = 1, 2 \ldots, 100$. The squared error of this prediction in pixel space was recorded, and averaged over all the different filtering extents t_1 to obtain means which are plotted in Figure 6.5(C). As baselines, we plot the error obtained by using the mean of filtered data as a predictor (titled 'Mean'), and the error obtained by using the last filtered observation as a predictor (titled 'Last').

Both baselines perform worse than any of the more complex algorithms (though as expected, the 'Last' predictor is a good one-step predictor), indicating that this is a non-trivial prediction problem. The LDS does well initially (due to smoothness), and the HMM does well in the longer run (due to competitive inhibition), while the RR-HMM performs as well or better at both time scales since it models both the smooth state evolution and competitive inhibition in its predictive distribution. In particular, the RR-HMM yields significantly lower prediction error consistently for the first 30 timesteps (i.e. five seconds) of the prediction horizon.

6.4 Related Work

6.4.1 Predictive State Representations

Predictive State Representations (PSRs) [68, 77] and Observable Operator Models (OOMs) [69] model sequence probabilities as a product of *observable operator* matrices. This idea, as well as the idea of learning such models using linear algebra techniques, originates in the literature on multiplicity automata and weighted automata [71, 72, 70]. Despite recent improvements [79, 80], practical learning algorithms for PSRs and OOMs have been lacking. RR-HMMs and its spectral learning algorithm are also closely related to methods in *subspace identification* [27, 29] in control systems for learning LDS parameters, which use SVD to determine the relationship between hidden states and observations.

As pointed out earlier, the spectral learning algorithm presented here learns PSRs. We briefly discuss other algorithms for learning PSRs from data. Several learning algorithms for PSRs have been proposed [81, 75, 82]. It is easier for PSR learning algorithms to return *consistent* parameter estimates because the parameters are based on observable quantities. [74] develops an SVD-based method for finding a low-dimensional variant of PSRs, called *Transformed PSRs* (TPSRs). Instead of tracking the probabilities of a small number of tests, TPSRs track a small number of linear combinations of a larger number of tests. This allows more compact representations, as well as dimensionality selection based on examining the singular values of the decomposed matrix, as in subspace identification methods. Note that nonlinearity can be encoded into the design of core tests. [83] introduced the concept of *e-tests* in PSRs that are indicator functions of aggregate sets of future outcomes, e.g. all sequence of observations in the immediate future that end with a particular observation after k timesteps. In general, tests in discrete PSRs can be indicator functions of arbitrary statistics of future events, thus encoding nonlinearities that might be essential for modeling some dynamical systems. Recently, Exponential Family PSRs (EFPSRs) [79] were introduced as an attempt to generalize the PLG model to allow general exponential family distributions over the next N observations. In the EFPSR, state is represented by modeling the parameters of a time-varying exponential family distribution over the next N timesteps. This allows graphical structure to be encoded in the distribution, by choosing the parameters accordingly. The justification for choosing an exponential family comes from maximum entropy modeling. Though inference and parameter learning are difficult in graphical models of non-trivial structure, approximate inference methods can be utilized to make these problems tractable. Like PLGs, the dynamical component of EFPSRs is modeled by *extending* and *conditioning* the distribution over time. However, the method presented [79] has some drawbacks, e.g. the extend-and-condition method is inconsistent with respect to marginals over individual timesteps between the extended and un-extended distributions.

6.4.2 Hybrid Models, Mixture Models and other recent approaches

RR-HMMs and their algorithms are also related to other *hybrid models*. Note that previous models of the same name (e.g. [84]) address a completely different problem, i.e. reducing the rank of the Gaussian observation parameters. Since shortly after the advent of LDSs, there have been attempts to combine the discrete states of HMMs with the smooth dynamics of LDSs. We perform a brief review of the literature on hybrid models; see [85] for a more thorough review. [86] formulates a switching LDS variant where both the state and observation variable noise models are mixture of Gaussians with the mixture switching variable evolving according to Markovian dynamics, and derives the (intractable) optimal filtering equations where the number of Gaussians needed to represent the belief increases exponentially over time. They also propose an approximate filtering algorithm for this model based on a single Gaussian. [87] proposes learning algorithms for an LDS with switching observation matrices. [88] reviews models where both the observations and state variable switch according to a discrete variable with Markov transitions. *Hidden Filter HMMs* (HFHMMs) [89] combine discrete and real-valued state variables and outputs that depend on both. The real-valued state is deterministically dependent on previous observations in a known manner, and only the discrete variable is hidden. This allows exact inference in this model to be tractable. [90] formulates the *Mixture Kalman Filter* (MKF) model along with a filtering algorithm, similar to [86] except that the filtering algorithm is based on sequential Monte-Carlo sampling.

The commonly used *HMMs with mixture-model observations* (e.g., Gaussian mixture) are a special case of RR-HMMs. A k-state HMM where each state corresponds to a Gaussian mixture of m observation models of n dimensions each is subsumed by a k-rank RR-HMM with m distinct continuous observations of n dimensions each, since the former is constrained to be non-negative and ≤ 1 in various places (the k-dimensional transition matrix, the k-dimensional belief vector, the matrix which transforms this belief to observation probabilities) where the latter is not.

Switching State-Space Models (SSSMs) [85] posit the existence of several real-valued hidden state variables that evolve linearly, with a single Markovian discrete-valued switching variable selecting the state which explains the real-valued observation at every timestep. Since exact inference and learning are intractable in this model, the authors derive a structured variational approximation that decouples the state space and switching variable chains, effectively resulting in Kalman smoothing on the state space variables and HMM forward-backward on the switching variable. In their experiments, the authors find SSSMs to perform better than regular LDSs on a physiological data modeling task with multiple distinct underlying dynamical models. HMMs performed comparably well in terms of log-likelihood, indicating their ability to model nonlinear dynamics though the resulting model was less interpretable than the best SSSM. More recently, models for nonlinear time series modeling such as Gaussian Process Dynamical Models have been proposed [91]. However, the parameter learning algorithm is only locally optimal, and exact inference and simulation are very expensive, requiring MCMC over a long sequence of frames all at once. This necessitates the use of heuristics for both inference and learning. Another recent nonlinear dynamic model is [92], which differs greatly from other methods in that it treats each component of the dynamic model learning problem separately using supervised learning algorithms, and proves consistency on the aggregate result under certain strong assumptions.

6.5 Discussion

Much as the Subspace ID algorithm (Chapter 3) provides a predictive, local optima-free method for learning continuous latent-variable models, the spectral learning algorithm here blurs the line between latent variable models and PSRs. PSRs were developed with a focus on the problem of an agent planning actions in a partially observable environment. More generally, there are many scenarios in sequential data modeling where the underlying dynamical system has inputs. The inference task for a learned model is then to track the belief state while conditioning on observations and incorporating the inputs. The input-output HMM (IO-HMM) [17] is a conditional probabilistic model which has these properties. A natural generalization of this work is to the task of learning RR-HMMs with inputs, or controlled PSRs. We discuss this further in Chapter 7. We recently carried out this generalization to controlled PSRs; details can be found in [73].

The question of proving containment or equivalence of RR-HMMs with respect to PSRs is of theoretical interest. The observable representation of an RR-HMM is a Transformed PSR (TPSR) [74], so every RR-HMM is a PSR; it remains to be seen whether every PSR corresponds to some RR-HMM (possibly with an infinite number of discrete hidden states) as well. The idea that "difficult" PSRs should somehow correspond to RR-HMMs with very large or infinite state space is intuitively appealing but not straightforward to prove. Another interesting direction would be to bound the performance of the learning algorithm when the underlying model is only approximately a reduced-rank HMM, much as the HKZ algorithm includes bounds when the underlying model is approximately an HMM [12]. This would be useful since in practice it is more realistic to expect any underlying system to not comply with the exact model assumptions.

The positive realization problem, i.e. obtaining stochastic transition and observation matrices from the RR-HMM observable representation, is also significant, though the observable representation allows us to carry out all possible HMM operations. HKZ describes a method based on [93] which, however, is highly erratic in practice. In the RR-HMM case, we have the additional challenge of firstly computing the minimal m for which a positive realization exists, and since the algorithm learns PSRs there is no guarantee that a particular set of learned parameters conforms exactly to any RR-HMM. On the applications side, it would be interesting to compare RR-HMMs with other dynamical models on classification tasks, as well as on learning models of difficult video modeling and graphics problems for simulation purposes. More elaborate choices of features may be useful in such applications, as would be the usage of high-dimensional or infinite-dimensional features via Reducing Kernel Hilbert Spaces (RKHS).

Chapter 7

Future Work and Discussion

The area of dynamical system modeling is of great importance to machine learning and robotics, and the ideas in this thesis lead to many promising avenues of research in these fields. In this chapter we first describe several possible extensions of models and algorithms we have presented. We next summarize the main contributions of this thesis and discuss its significance in the context of the larger body of machine learning research.

7.1 Future Work

7.1.1 Scaling STACS for learning very large state-space HMMs

While efficient *inference* algorithms have been proposed for HMMs with up to a thousand states based on specific properties of the underlying parameter space [48, 94], it is more difficult to *learn* exact HMMs of such large finite state space size. Research on the infinite HMM and its variants [22, 21] provides sampling-based methods for learning nonparametric HMM models with an effectively infinite number of states, but these algorithms are inefficient and don't scale well to high-dimensional sequences. The STACS and V-STACS algorithms presented in Chapter 4 efficiently learn accurate models of HMMs with up to a hundred states or so. Beyond that, several factors hinder efficient and accurate learning:

1. The computational cost of performing split-tests on each HMM state at every model expansion step.

2. The cost of performing EM or Viterbi Training on the entire HMM in every model learning step.

3. The increased chance of *false positives* in the candidate testing step.

The upshot is that, if we do manage to learn accurate models of this size, the aforementioned inference algorithms for large-state-space HMMs would allow us to carry out efficient inference in these learned HMMs under a variety of possible additional assumptions, some of which are relatively reasonable for large state spaces (e.g. assuming the transition matrix is additively sparse $+$ low-rank [48]).

One way to address (1) is *lazy testing* of candidates. If a particular candidate state changes minimally between model expansion steps in terms of its transition and observation distributions, it is unlikely that its rank as a split candidate will change much. We can develop this intuition further to form a criterion for selectively testing only those candidates that either were likely candidates in the previous model expansion step or have changed substantially since then. Another, more exact way of addressing (1) is to take advantage of the highly decoupled nature of split-tests for each state and run them in *parallel*, which is an increasingly feasible option in light of the growing availability of parallel architectures and software for parallelizing conventional machine learning algorithms.

The limitation of performing EM on large state spaces (2) can similarly be addressed by exploiting recent developments in parallelizing belief propagation (e.g. [95]). Another possibility is to only update regions of state space that have changed significantly w.r.t. previous iterations.

During split testing, the null hypothesis states that no split justifies the added complexity (either through BIC score, test-set log-likelihood or other metric). The increased risk of false positives in split testing (3) arises from the increased number of alternative hypotheses available, since the likelihood that one of them will randomly beat the null hypothesis increases. We can use techniques from *multiple hypothesis testing* [96] to modify the split scores and compensate for this phenomenon. Though this avoids continuing to split needlessly, it doesn't necessarily make sure we choose the best candidate at each iteration. A more accurate scoring criterion based on variational Bayes [58] or using test-set loglikelihood for scoring (if enough data and computational resources are present) would help address this issue.

7.1.2 Constraint generation for learning stable PSRs

Stability is a desirable characteristic of a wide variety of sequential data models. For example, a necessary condition for a PSR to be stable is that the sum of the observable operators in a PSR or OOM must have at most unit spectral radius.[1] Chapter 6 describes a spectral learning algorithm for RR-HMMs that is also shown to be able to learn PSRs/OOMs (Section A.5 in the Appendix). The constraint generation algorithm described in Chapter 5 can be adapted to solve a quadratic programming problem that constrains the appropriate spectral radius in order to achieve stability. Since PSRs with *actions* (i.e. *controlled PSRs*) were designed with the goal of enabling efficient and accurate *planning*, stability becomes especially desirable for PSR parameters.

7.1.3 Efficient planning in empirically estimated RR-POMDPs

Because of the difficulties associated with learning POMDPs from data, POMDP planning is typically carried out using hand-coded POMDPs whose belief evolves in a pre-specified state space. However, the spectral learning algorithm for RR-HMMs, together with efficient point-based planning algorithms such as [98, 99, 100], can allow us to *close the loop* by learning models, planning in them and using the resultant data to update the learnt model. This effectively leaves the task of state space formulation up to spectral learning to decide the optimal subspace for planning. While not as easily interpretable *a priori*, this method has the potential of discovering far more compact state spaces for planning prob-

[1]the sufficient condition, namely that the model only emits positive probabilities, is undecidable [97]. However, setting negatives to a small positive number and renormalizing works in practice.

lems based on the specific task and value function involved, especially since RR-POMDPs are more expressive than POMDPs of the same dimension. The connection of RR-HMMs to PSRs means that such a generalization would effectively learn models of controlled PSRs from data. We have recently extended the spectral learning algorithm to this scenario, with positive results; in particular, we are able to close the learning-planning loop by carrying out planning in a model estimated from data on a simulated high-dimensional vision-based robot navigation task [73]. Note that this is different from Linear-Quadratic-Gaussian (LQG) control, which is the task of optimally controlling an LDS that has control inputs.

7.1.4 Learning RR-HMMs over arbitrary observation features

We discussed how to learn RR-HMMs over single observations as well as stacked vectors of multiple observations for multiple-step observable systems. The logical generalization is to enable RR-HMMs over arbitrary *indicative and characteristic features* of past and future observations. We can show that the same spectral learning algorithm yields unbiased estimates for RR-HMM parameters over such features. Generalization of the sample complexity bounds should in principle follow using the same sampling bounds and matrix perturbation bounds as used here, and is an important task for future work.

7.1.5 Hilbert space embeddings of RR-HMMs

An alternative generalization of RR-HMM expressiveness is to infinite-dimensional feature spaces via kernels. Recent work has shown how to embed conditional distributions in Hilbert space [101], allowing generalization of conditional distributions to infinite-dimensional feature spaces for several applications. Since a sequential model is basically a series of conditional distributions, [101] describes an application of conditional distribution embedding to learning LDS models when the latent state is observed. It should be possible to formulate a kernelized HMM or RR-HMM in a similar fashion. Further extending the spectral learning method to learn RR-HMMs in Hilbert space would allow

HMMs over much more complex observation spaces.

7.1.6 Sample complexity bounds for spectral learning of PSRs

We saw in Chapter 6 that the RR-HMM learning algorithm returns unbiased estimates of PSRs as well. In fact, even though for finite data samples the model returned may not correspond to any RR-HMM of finite state space size, it always corresponds to some PSR if it is stable. What remains to be done is to generalize the sample complexity bounds to the PSR case as well. Since there are no distinct observation and transition matrices in PSRs, and the observable operators and belief states are not necessarily stochastic, the bounds and proofs may differ. However, the same basic perturbation and sampling error ideas should, in theory, hold.

7.1.7 Spectral learning of exponential family RR-HMMs

Belief Compression [102] exploits the stochasticity of HMM beliefs in order to compress them more effectively for POMDP planning using exponential family PCA [103]. In a similar vein, it should be possible to extend RR-HMMs and their learning algorithms to the exponential family case. Gordon (2002)[104] presents a more efficient and general algorithm for carrying out exponential family PCA which could be used for such an extension.

7.2 Why This Thesis Matters

Whether the goal is to perform accurate prediction and sequence classification for statistical data mining, or to model a partially observable dynamic environment for reasoning under uncertainty, the modeling of dynamical systems is a central task. Due to their theoretical properties and their effectiveness in practice, LVMs are the predominant class of probabilistic models for representing dynamical systems. This thesis has proposed several new algorithms for learning LVMs from data more efficiently and accurately than previous methods, and has also proposed a novel model that combines desirable properties of existing models and can be learned with strong performance guarantees. The thesis contributes a number of novel ideas for addressing fundamental problems in sequential data modeling techniques. For example, we address the problems of *model selection and local minima* in EM-based HMM learning by proposing the first practical top-down HMM model selection algorithm that discovers new states using properties of the observation distribution as well as the temporal patterns in the data. We also address the issue of *instability*, proposing a constraint-generation algorithm that outperforms previous methods in both efficiency and accuracy. Finally, the RR-HMM is a hybrid model that combines the discrete spaces and competitive inhibition of HMMs with the smooth state evolution of LDSs. The spectral learning algorithm we propose for it is qualitatively different from EM-based methods, and is based on the first known HMM learning algorithm with provable performance guarantees. We extended the bounds in a way that makes them significantly tighter for cases where the actual HMM rank is much lower than its state space size. This model and its learning algorithm also blurs the line of distinction between LVMs and PSRs even further.

For each of the above three branches of this thesis we demonstrated experimental results on a wide variety of real-world data sets. We also outlined numerous promising extensions of all three branches earlier in this chapter. The research presented in this thesis has the potential to positively impact a variety of fields of application where sequential data modeling is important, such as robot sensing and planning, video modeling in computer vision, activity recognition and user modeling, speech recognition, time series forecasting in science and science, bioinformatics, and more. Many of the extensions described above are already being researched and implemented, demonstrating that this thesis lays the groundwork for several fruitful lines of current and future research in the field of statistical machine learning as well as in application areas such as robotics.

Appendix A

RR-HMM Details

This appendix contains details omitted from Chapter 6 including learning algorithm performance proof details, a proof sketch regarding learning with ambiguous observations, and parameter values for synthetic HMM experiments.

A.1 Proofs

Proofs for theoretical guarantees on the RR-HMM learning algorithm are given below, after some preliminary conditions and lemmas.
The proof of Theorem 2 relies on Lemmas 18 and 24. We start off with some preliminary results and build up to proving the main theorem and its lemmas below.

A remark on norms: The notation $\|X\|_p$ for *matrices* $X \in \mathbb{R}^{m\times n}$ denotes the *operator norm* $\max \frac{\|Xv\|_p}{\|v\|_p}$ for vector $v \neq 0$. Specifically, $\|X\|_2$ denotes L_2 *matrix norm* (also known as *spectral norm*), which corresponds to the *largest singular value* $\sigma_1(X)$. *Frobenius norm* is denoted by $\|X\|_F = \left(\sum_{i=1}^m \sum_{j=1}^n X_{ij}^2\right)^{1/2}$. The notation $\|X\|_1$ for matrices denotes the L_1 matrix norm which corresponds to *maximum absolute column sum* $\max_c \sum_{i=1}^m |X_{ic}|$. The definition of $\|x\|_p$ for *vectors* $x \in \mathbb{R}^n$ is the standard distance measure $(\sum_{i=1}^n x_i^p)^{1/p}$.

A.1.1 Preliminaries

The following conditions are assumed by the main theorems and algorithms.

Condition 3 *[Modification of HKZ Condition 1] $\vec{\pi} > 0$ element-wise, T has rank k (i.e. R and S both have rank k) and O has rank at least k.*

The following two conditions on R can always be satisfied by scaling down entries in R and scaling up S accordingly. However we want entries in R to be as large as possible under the two conditions below, so that $\sigma_k(U^\top OR)$ is large and $1/\sigma_k(U^\top OR)$ is small to make the error bound as tight as possible (Theorem 2). Hence we pay for scaling down R by loosening the error bound we obtain for a given number of training samples.

Condition 4 $\|R\|_1 \leq 1$.

Condition 5 *For some column $1 \leq c \leq k$ of R, it is the case that $\|R[\cdot, c]\|_2 \leq \sqrt{k/m}$.*

The above two conditions on R ensure the bounds go through largely unchanged from HKZ aside from the improvement due to low rank k. The first condition can be satisfied in a variety of ways without loss of generality, e.g. by choosing the columns of R to be any k independent columns of T, and S to be the coefficients needed to reconstruct T from R. Intuitively, the first condition implies that R does not overly magnify the magnitude of vectors it multiplies with. The second one implies a certain degree of uniformity in at least one of the columns of R. For example, the uniform distribution in a column of R would satisfy the constraint, whereas a column of the identity matrix would not. This does not imply that T must have a similarly near-uniform column. We can form R from the uniform distribution along with some independent columns of T.

The observable representation depends on a matrix $U \in \mathbb{R}^{n \times k}$ that obeys the following condition:

Condition 6 *[Modification of HKZ Condition 2] $U^\top OR$ is invertible.*

This is analogous to the HKZ invertibility condition on $U^\top O$, since OR is the matrix that yields observation probabilities from a *low-dimensional* state vector. Hence, U defines a k-dimensional subspace that preserves the *low-dimensional* state dynamics regardless of the number of states m.

Condition 7 *Assume that $S\ \mathrm{diag}(\vec{\pi})O^\top$ has full row rank (i.e. k).*

This condition amounts to ensuring that that the ranges S and O, which are both at least rank k, overlap enough to preserve the dynamics when mapping down to the low-dimensional state. As in HKZ, the left singular vectors of $P_{2,1}$ give us a valid U matrix.

Lemma 8 *[Modification of HKZ Lemma 2] Assume Conditions 3 and 7. Then,* $\mathrm{rank}(P_{2,1}) = k$*. Also, if U is the matrix of left singular vectors of $P_{2,1}$ corresponding to non-zero singular values, then* $\mathrm{range}(U) = \mathrm{range}(OR)$*, so $U \in \mathbb{R}^{n\times k}$ obeys Condition 6.*

PROOF: From its definition, we can show $P_{2,1}$ can be written as a low-rank product of RR-HMM parameters:

$$\begin{aligned}
[P_{2,1}]_{i,j} &= \Pr[x_2 = i, x_1 = j] \\
&= \sum_{a=1}^{m}\sum_{b=1}^{m} \Pr[x_2 = i, x_1 = j, h_2 = a, h_1 = b] \quad \text{(marginalizing hidden states } h\text{)} \\
&= \sum_{a=1}^{m}\sum_{b=1}^{m} \Pr[x_2 = i | h_2 = a]\Pr[h_2 = a | h_1 = b]\Pr[x_1 = j | h_1 = b]\Pr[h_1 = b] \\
&= \sum_{a=1}^{m}\sum_{b=1}^{m} O_{ia}T_{ab}\vec{\pi}_b[O^\top]_{bj} \\
\Rightarrow P_{2,1} &= OT\ \mathrm{diag}(\vec{\pi})O^\top \\
&= ORS\ \mathrm{diag}(\vec{\pi})O^\top \qquad \text{(A.1)}
\end{aligned}$$

Thus $\mathrm{range}(P_{2,1}) \subseteq \mathrm{range}(OR)$. This shows that $\mathrm{rank}(P_{2,1}) \leq \mathrm{rank}(OR)$.

By Condition 7, $S\ \mathrm{diag}(\vec{\pi})O^\top$ has full row rank, thus $S\ \mathrm{diag}(\vec{\pi})O^\top(S\ \mathrm{diag}(\vec{\pi})O^\top)^+ = I_{k\times k}$. Therefore,

$$OR = P_{2,1}(S\ \mathrm{diag}(\vec{\pi})O^\top)^+ \tag{A.2}$$

which implies $\mathrm{range}(OR) \subseteq \mathrm{range}(P_{2,1})$, which in turn implies $\mathrm{rank}(OR) \leq \mathrm{rank}(P_{2,1})$.

Together this proves that $\mathrm{rank}(P_{2,1}) = \mathrm{rank}(OR)$, which we can show to be k as follows: Condition 3 implies that $\mathrm{rank}\left(U^\top OR\right) = k$, and hence $\mathrm{rank}(OR) \geq k$. Since $OR \in \mathbb{R}^{m\times k}$, $\mathrm{rank}(OR) \leq k$. Therefore, $\mathrm{rank}(OR) = k$. Hence $\mathrm{rank}(P_{2,1}) = k$.

Since $\mathrm{range}(U) = \mathrm{range}(P_{2,1})$ by definition of singular vectors, this implies $\mathrm{range}(U) = \mathrm{range}(OR)$. Therefore, $(U^\top OR)$ is invertible and hence U obeys Condition 6. $\square$

The following lemma shows that the observable representation $\{\vec{b}_\infty, \vec{b}_1, B_1, \ldots, B_n\}$ is linearly related to the true HMM parameters, and can compute the probability of any sequence of observations.

Lemma 9 *[Modification of HKZ Lemma 3] (Observable HMM Representation). Assume Condition 3 on the RR-HMM and Condition 6 on the matrix $U \in \mathbb{R}^{n\times k}$. Then, the observable representation of the RR-HMM (Definition 1 of the paper) has the following properties:*

1. $\vec{b}_1 \quad = \quad (U^\top OR)\pi_l \quad = \quad (U^\top O)\pi$,

2. $\vec{b}_\infty^\top \quad = \quad 1_m^\top R(U^\top OR)^{-1}$,

3. *For all* $x = 1, \ldots, n :\ B_x \quad = \quad (U^\top OR)W_x(U^\top OR)^{-1}$

4. *For any time* t: $\Pr[x_{1:t}] \quad = \quad \vec{b}_\infty^\top B_{x_{t:1}}\vec{b}_1$

PROOF:

1. We can write $\vec{P}_1$ as $O\pi$, since

$$\begin{aligned}[\vec{P}_1]_i &= \Pr[x_1 = i] \\ &= \sum_{a=1}^{m} \Pr[x_1 = i | h_1 = a] \Pr[h_1 = a] \\ &= \sum_{a=1}^{m} O_{ia} \vec{\pi}_a\end{aligned}$$

   Combined with the fact that $\vec{b}_1 = U^\top \vec{P}_1$ by definition, this proves the first claim.

2. Firstly note that $\vec{P}_1^\top = \vec{1}_m^\top T \ \mathrm{diag}(\vec{\pi}) O^\top$, since

$$\begin{aligned}\vec{P}_1^\top &= \vec{\pi}^\top O^\top \\ &= \vec{1}_m^\top \ \mathrm{diag}(\vec{\pi}) O^\top \\ &= \vec{1}_m^\top T \ \mathrm{diag}(\vec{\pi}) O^\top \quad (\text{since } \vec{1}_m^\top T = \vec{1}_m^\top)\end{aligned}$$

   This allows us to write $\vec{P}_1$ in the following form:

$$\begin{aligned}\vec{P}_1^\top &= \vec{1}_m^\top T \ \mathrm{diag}(\pi) O^\top \\ &= \vec{1}_m^\top RS \ \mathrm{diag}(\pi) O^\top \\ &= \vec{1}_m^\top R(U^\top OR)^{-1}(U^\top OR) S \ \mathrm{diag}(\pi) O^\top \\ &= \vec{1}_m^\top R(U^\top OR)^{-1} U^\top P_{2,1} \quad (\text{by equation (A.1)})\end{aligned}$$

   By equation (A.1), $U^\top P_{2,1} = (U^\top OR) S \ \mathrm{diag}(\vec{\pi}) O^\top$. Since $(U^\top OR)$ is invertible by Condition 6, and $S \ \mathrm{diag}(\vec{\pi}) O^\top$ has full row rank by Condition 7, we know that $(U^\top P_{2,1})^+$ exists and

$$U^\top P_{2,1} (U^\top P_{2,1})^+ = I_{k \times k} \tag{A.3}$$

   Therefore,

$$b_\infty^\top = P_1^\top (U^\top P_{2,1})^+ = 1_m^\top R(U^\top OR)^{-1} (U^\top P_{2,1})(U^\top P_{2,1})^+ = 1_m^\top R(U^\top OR)^{-1}$$

   hence proving the second claim.

3. The third claim can be proven by first expressing $P_{3,x,1}$ as a product of RR-HMM parameters:

$$\begin{aligned}
[P_{3,x,1}]_{ij} &= \Pr[x_3 = i, x_2 = x, x_1 = j] \\
&= \sum_{a=1}^{m}\sum_{b=1}^{m}\sum_{c=1}^{m} \Pr[x_3 = i, x_2 = x, x_1 = j, h_3 = a, h_2 = b, h_1 = c] \\
&= \sum_{a=1}^{m}\sum_{b=1}^{m}\sum_{c=1}^{m} \Pr[x_3 = i | h_3 = a] \Pr[h_3 = a | h_2 = b] \Pr[x_2 = x | h_2 = b] \\
&\qquad \Pr[h_2 = b | h_1 = c] \Pr[h_1 = c] \Pr[x_1 = j | h_1 = c] \\
&= \sum_{a=1}^{m}\sum_{b=1}^{m}\sum_{c=1}^{m} O_{ia}[A_x]_{ab}T_{bc}\vec{\pi}_c[O^\mathsf{T}]_{cj} \\
\Rightarrow P_{3,x,1} &= OA_xT \ \mathrm{diag}(\vec{\pi})O^\mathsf{T}
\end{aligned}$$

This can be transformed as follows:

$$\begin{aligned}
P_{3,x,1} &= OA_xRS \ \mathrm{diag}(\vec{\pi})O^\mathsf{T} \\
&= OA_xR(U^\mathsf{T}OR)^{-1}(U^\mathsf{T}OR)S \ \mathrm{diag}(\vec{\pi})O^\mathsf{T} \\
&= OA_xR(U^\mathsf{T}OR)^{-1}U^\mathsf{T}(OT \ \mathrm{diag}(\vec{\pi})O^\mathsf{T}) \\
&= OA_xR(U^\mathsf{T}OR)^{-1}U^\mathsf{T}P_{2,1} \quad \text{(by equation (A.1))}
\end{aligned}$$

and then plugging in this expression into the definition of B_x, we obtain the required result:

$$\begin{aligned}
B_x &= (U^\mathsf{T}P_{3,x,1})(U^\mathsf{T}P_{2,1})^+ \quad \text{(by definition)} \\
&= (U^\mathsf{T}O)A_xR(U^\mathsf{T}OR)^{-1}(U^\mathsf{T}P_{2,1})(U^\mathsf{T}P_{2,1})^+ \\
&= (U^\mathsf{T}O)A_xR(U^\mathsf{T}OR)^{-1} \quad \text{(by equation (A.3))} \\
&= (U^\mathsf{T}OR)\left(S \ \mathrm{diag}(O_{x,\cdot})R\right)(U^\mathsf{T}OR)^{-1} \\
&= (U^\mathsf{T}OR)W_x(U^\mathsf{T}OR)^{-1}
\end{aligned}$$

4. Using the above three results, the fourth claim follows from equation 1 in Section 2 in the paper:

$$\begin{aligned}
&\Pr[x_1, \ldots, x_t] \\
&= \vec{1}_m^\top R W_{x_t} \ldots W_{x_1} \vec{\pi}_l \\
&= \vec{1}_m^\top R (U^\top OR)^{-1} (U^\top OR) W_{x_t} (U^\top OR)^{-1} (U^\top OR) W_{x_{t-1}} (U^\top OR)^{-1} \ldots \\
&\quad \ldots (U^\top OR) W_{x_1} (U^\top OR)^{-1} (U^\top OR) \vec{\pi}_l \\
&= \vec{b}_\infty^\top B_{x_t} \ldots B_{x_1} \vec{b}_1 \\
&= \vec{b}_\infty^\top B_{x_{t:1}} \vec{b}_1
\end{aligned}$$

□

In addition to $\vec{b}_1$ above, we define normalized conditional 'internal states' $\vec{b}_t$ that help us compute conditional probabilities. These internal states are not probabilities. In contrast to HKZ where these internal states are m-dimensional vectors, in our case the internal states are k-dimensional i.e. they correspond to the rank of the HMM. As shown above in Lemma 9,

$$\vec{b}_1 = (U^\top OR)\vec{\pi}_l = (U^\top O)\vec{\pi}$$

In addition for any $t \geq 1$, given observations $x_{1:t-1}$ with non-zero probability, the internal state is defined as:

$$\vec{b}_t = \vec{b}_t(x_{1:t-1}) = \frac{B_{x_{t-1:1}} \vec{b}_1}{\vec{b}_\infty^\top B_{x_{t-1:1}} \vec{b}_1} \tag{A.4}$$

For $t = 1$ the formula is still consistent since $\vec{b}_\infty^\top \vec{b}_1 = \vec{1}_m^\top R (U^\top OR)^{-1} (U^\top OR) \vec{\pi}_l = \vec{1}_m^\top R \vec{\pi}_l = \vec{1}_m^\top \vec{\pi} = 1$.

Recall that HMM and RR-HMM parameters can be used to calculate joint probabilities as follows:

$$\begin{aligned}
\Pr[x_1, ..., x_t] &= \vec{1}_m^\top A_{x_t} A_{x_{t-1}} \cdots A_{x_1} \vec{\pi} \\
&= \vec{1}_m^\top RS \ \text{diag}(O_{x_t,\cdot}) RS \ \text{diag}(O_{x_{t-1},\cdot}) R \cdots S \ \text{diag}(O_{x_1,\cdot}) \vec{\pi} \\
&= \vec{1}_m^\top R \left(S \ \text{diag}(O_{x_t,\cdot}) R\right) \left(S \ \text{diag}(O_{x_{t-1},\cdot}) R\right) \cdots \left(S \ \text{diag}(O_{x_1,\cdot}) R\right) S\vec{\pi} \\
&= \vec{1}_m^\top R W_{x_t} \ldots W_{x_1} \vec{\pi}_l \quad \text{(by definition of } W_x, \vec{\pi}_l\text{)}
\end{aligned} \tag{A.5}$$

The following Lemma shows that the conditional internal states are linearly related, and also shows how we can use them to compute conditional probabilities.

Lemma 10 *[Modification of HKZ Lemma 4] (Conditional Internal States) Assume the conditions of Lemma 9 hold, i.e. Conditions 3 and 6 hold. Then, for any time t:*

1. *(Recursive update) If $\Pr[x_1, \ldots, x_t] > 0$, then*
$$\vec{b}_{t+1} = \frac{B_{x_t}\vec{b}_t}{\vec{b}_\infty^\top B_{x_t}\vec{b}_t}$$

2. *(Relation to hidden states)*
$$\vec{b}_t = (U^\top OR)l_t(x_{1:t-1}) = (U^\top O)h_t(x_{1:t-1})$$
where $[\vec{h}_t(x_{1:t-1})]_i = \Pr[h_t = i|x_{1:t-1}]$ is defined as the conditional probability of the hidden state at time t given observations $x_{1:t-1}$, and $\vec{l}_t(x_{1:t-1})$ is its low-dimensional projection such that $\vec{h}_t(x_{1:t-1}) = R\vec{l}_t(x_{1:t-1})$.

3. *(Conditional observation probabilities)*
$$\Pr[x_t|x_{1:t-1}] = \vec{b}_\infty^\top B_{x_t}\vec{b}_t$$

PROOF: The first proof is direct, the second follows by induction.

1. The $t = 2$ case $\vec{b}_2 = \frac{B_{x_1}\vec{b}_1}{\vec{b}_\infty^\top B_{x_1}\vec{b}_1}$ is true by definition (equation A.4). For $t \geq 3$, again by definition of b_{t+1} we have
$$\begin{aligned}
\vec{b}_{t+1} &= \frac{B_{x_{t:1}}\vec{b}_1}{\vec{b}_\infty^\top B_{x_{t:1}}\vec{b}_1} \\
&= \frac{B_{x_t}B_{x_{t-1:1}}\vec{b}_1}{\frac{\vec{b}_\infty^\top B_{x_{t-1:1}}\vec{b}_1}{\vec{b}_\infty^\top B_{x_{t-1:1}}\vec{b}_1}\vec{b}_\infty^\top B_{x_t}B_{x_{t-1:1}}\vec{b}_1} \\
&= \frac{B_{x_t}\vec{b}_t}{\vec{b}_\infty^\top B_{x_t}\frac{B_{x_{t-1:1}}\vec{b}_1}{\vec{b}_\infty^\top B_{x_{t-1:1}}\vec{b}_1}} \quad \text{(by equation (A.4))} \\
&= \frac{B_{x_t}\vec{b}_t}{\vec{b}_\infty^\top B_{x_t}\vec{b}_t} \quad \text{(by equation (A.4))}
\end{aligned}$$

2,3. The base case for claim 2 holds by Lemma 9, since $\vec{h}_1 = \vec{\pi}$, $\vec{l}_1 = R\vec{\pi}$ and $\vec{b}_1 = (U^\top OR)\vec{\pi}$. For claim 3, the base case holds since $\vec{b}_\infty^\top B_{x_1}\vec{b}_1 = \vec{1}_m^\top RW_{x_1}\vec{\pi}_l$ by Lemma 9, which equals $\Pr[x_1]$ by equation (A.5). The inductive step is:

$$\begin{aligned}
\vec{b}_{t+1} &= \frac{B_{x_t}\vec{b}_t}{\vec{b}_\infty^\top B_{x_t}\vec{b}_t} \quad \text{(by claim 1 above)} \\
&= \frac{B_{x_t}(U^\top OR)\vec{l}_t}{\Pr[x_t|x_{1:t-1}]} \quad \text{(by inductive hypothesis)} \\
&= \frac{(U^\top OR)W_{x_t}\vec{l}_t}{\Pr[x_t|x_{1:t-1}]} \quad \text{(by Lemma 9)} \\
&= \frac{(U^\top O)A_{x_t}\vec{h}_t}{\Pr[x_t|x_{1:t-1}]} \quad (\because RW_{x_t}\vec{l}_t = RS\ \operatorname{diag}(O_{x_t,\cdot})R\vec{l}_t = A_{x_t}\vec{l}_t)
\end{aligned}$$

Now by definition of $A_{x_t}\vec{h}_t$,

$$\begin{aligned}
\vec{b}_{t+1} &= (U^\top O)\frac{\Pr[h_{t+1} = \cdot, x_t|x_{1:t-1}]}{\Pr[x_t|x_{1:t-1}]} \\
&= (U^\top O)\frac{\Pr[h_{t+1} = \cdot|x_{1:t}]\Pr[x_t|x_{1:t-1}]}{\Pr[x_t|x_{1:t-1}]} \\
&= (U^\top O)\vec{h}_{t+1}(x_{1:t}) \\
&= (U^\top OR)\vec{l}_{t+1}(x_{1:t})
\end{aligned}$$

This proves claim 2, using which we can complete the proof for claim 3:

$$\begin{aligned}
\vec{b}_\infty^\top B_{x_{t+1}}\vec{b}_{t+1} &= \vec{1}_m^\top R(U^\top OR)^{-1}(U^\top OR)W_{x_t}(U^\top OR)^{-1}\vec{b}_{t+1} \quad \text{(by Lemma 9)} \\
&= \vec{1}_m^\top RW_{x_t}(U^\top OR)^{-1}(U^\top OR)\vec{l}_{t+1} \quad \text{(by claim 2 above)} \\
&= \vec{1}_m^\top RW_{x_t}\vec{l}_{t+1} \\
&= \vec{1}_m^\top RS\ \operatorname{diag}(O_{x_t,\cdot})R\vec{l}_{t+1} \quad \text{(by definition of } W_{x_t}) \\
&= \vec{1}_m^\top T\ \operatorname{diag}(O_{x_t,\cdot})\vec{h}_{t+1} \\
&= \vec{1}_m^\top A_{x_t}\vec{h}_{t+1}
\end{aligned}$$

Again by definition of $A_{x_t}\vec{h}_{t+1}$,

$$\begin{aligned}\vec{b}_\infty^\top B_{x_{t+1}}\vec{b}_{t+1} &= \sum_{a=1}^{m}\sum_{b=1}^{m}\Pr[x_{t+1}|h_{t+1}=a]\Pr[h_{t+1}=a|h_t=b]\Pr[h_t=b|x_{1:t}]\\ &= \sum_{a=1}^{m}\sum_{b=1}^{m}\Pr[x_{t+1},h_{t+1}=a,h_t=b|x_{1:t}]\\ &= \Pr[x_{t+1}|x_{1:t}]\end{aligned}$$

□

Remark 11 *If U is the matrix of left singular vectors of $P_{2,1}$ corresponding to non-zero singular values, then U is the observable-representation analogue of the observation probability matrix O in the sense that, given a conditional state $\vec{b}_t$, $\Pr[x_t = i|x_{1:t-1}] = [U\vec{b}_t]_i$ in the same way as $\Pr[x_t = i|x_{1:t-1}] = [O\vec{h}_t]_i$ for a conditional hidden state $\vec{h}_t$.*

PROOF: Since $\text{range}(U) = \text{range}(OR)$ (Lemma 2), and $UU^\top$ is a projection operator to $\text{range}(U)$, we have $UU^\top OR = OR$, so $U\vec{b}_t = U(U^\top OR)l_t = ORl_t = Oh_t$. □

A.1.2 Matrix Perturbation Theory

We take a diversion to matrix perturbation theory and state some standard theorems from Steward and Sun (1990) [105] and Wedin (1972) [106] which we will use, and also prove a result from these theorems. The following lemma bounds the L_2-norm difference between the pseudoinverse of a matrix and the pseudoinverse of its perturbation.

Lemma 12 *(Theorem 3.8 of Stewart and Sun (1990) [105]) Let $A \in \mathbb{R}^{m\times n}$, with $m \geq n$, and let $\widetilde{A} = A + E$. Then,*

$$\left\|\widetilde{A}^+ - A^+\right\|_2 \leq \frac{1+\sqrt{5}}{2}\cdot\max\left\{\left\|A^+\right\|_2^2, \left\|\widetilde{A}^+\right\|_2^2\right\}\|E\|_2 \quad .$$

The following lemma bounds the absolute differences between the singular values of a matrix and its perturbation.

Lemma 13 *(Theorem 4.11 of Stewart and Sun (1990) [105]). Let $A \in \mathbb{R}^{m\times n}$ with $m \geq n$, and let $\widetilde{A} = A + E$. If the singular values of A and $\widetilde{A}$ are $(\sigma_1 \geq \ldots \geq \sigma_n)$ and $(\widetilde{\sigma}_1 \geq \ldots \geq \widetilde{\sigma}_n)$, respectively, then*

$$|\widetilde{\sigma}_i - \sigma_i| \leq \|E\|_2 \quad i = 1, \ldots, n$$

Before the next lemma we must define the notion of *canonical angles* between two subspaces:

Definition 14 *(Adapted from definition 4.35 of Stewart (1998) [107]) Let X and Y be matrices whose columns comprise orthonormal bases of two p-dimensional subspaces $\mathcal{X}$ and $\mathcal{Y}$ respectively. Let the singular values of $X^{\mathsf{T}}Y$ (where X^{T} denotes the conjugate transpose, or Hermitian, of matrix X) be $\gamma_1, \gamma_2, \ldots, \gamma_p$. Then the* canonical angles *θ_i between $\mathcal{X}$ and $\mathcal{Y}$ are defined by*

$$\theta_i = \cos^{-1}\gamma_i, \quad i = 1, 2, \ldots, p$$

The matrix of canonical angles Θ *is defined as*

$$\Theta(\mathcal{X}, \mathcal{Y}) = \operatorname{diag}(\theta_1, \theta_2, \ldots, \theta_p)$$

Note that $\forall i \;\; \gamma_i \in [0, 1]$ in the above definition, since γ_1 (assuming it's the highest singular value) is no greater than $\sigma_1(X^{\mathsf{T}})\sigma_1(Y) \leq 1 \cdot 1 = 1$, and hence $\cos^{-1}\gamma_i$ is always well-defined.

For any matrix A, define $A_\perp$ to be the *orthogonal complement* of the subspace spanned by the columns of A. For example, any subset of left singular vectors of a matrix comprise the orthogonal complement of the matrix composed of the remaining left singular vectors. The following lemma gives us a convenient way of calculating the sines of the canonical angles between two subspaces using orthogonal complements:

Lemma 15 *(Theorem 4.37 of Stewart (1998) [107]) Let X and Y be $n \times p$ matrices, with $n > p$, whose columns comprise orthonormal bases of two p-dimensional subspaces $\mathcal{X}$ and $\mathcal{Y}$ respectively. Assume $X_\perp, Y_\perp \in \mathbb{R}^{n\times n-p}$ such that $[X \;\, X_\perp]$ and $[Y \;\, Y_\perp]$ are* *orthogonal matrices. The singular values of $Y_\perp^\mathsf{T} X$ are the sines of the canonical angles between $\mathcal{X}$ and $\mathcal{Y}$.*

The following lemma bounds the L_2-norm difference between the sine of the canonical angle matrices of the range of a matrix and its perturbation.

Lemma 16 *([106],Theorem 4.4 of Stewart and Sun (1990) [105]). Let $A \in \mathbb{R}^{m\times n}$ with $m \geq n$, with the singular value decomposition ($U_1, U_2, U_3, \Sigma_1, \Sigma_2, V_1, V_2$):*

$$\begin{bmatrix} U_1^\mathsf{T} \\ U_2^\mathsf{T} \\ U_3^\mathsf{T} \end{bmatrix} A \begin{bmatrix} V_1 & V_2 \end{bmatrix} = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \\ 0 & 0 \end{bmatrix}$$

Let $\widetilde{A} = A + E$, with analogous SVD ($\widetilde{U}_1, \widetilde{U}_2, \widetilde{U}_3, \widetilde{\Sigma}_1, \widetilde{\Sigma}_2, \widetilde{\Sigma}_3, \widetilde{V}_1, \widetilde{V}_2$). Let Φ be the matrix of canonical angles between $\mathrm{range}(U_1)$ and $\mathrm{range}(\widetilde{U}_1)$, and Θ be the matrix of canonical angles between $\mathrm{range}(V_1)$ and $\mathrm{range}(\widetilde{V}_1)$. If there exists $\delta > 0, \alpha \geq 0$ such that $\min \sigma(\widetilde{\Sigma}_1) \geq \alpha + \delta$ and $\max \sigma(\Sigma_2) \leq \alpha$, then

$$\max \left\{ \left\| \sin \Phi \right\|_2, \left\| \sin \Theta \right\|_2 \right\} \leq \frac{\left\| E \right\|_2}{\delta}$$

The above two lemmas can be adapted to prove that Corollary 22 of HKZ holds for the low-rank case as well, assuming that the perturbation is bounded by a number less than σ_k. The following lemma shows that (1) the k^{th} singular value of a matrix and its perturbation are close to each other, and (2) that the subspace spanned by the first k singular vectors of a matrix is nearly orthogonal to the subspace spanned by the $(k+1)^{\mathrm{th}}, \ldots, m^{\mathrm{th}}$ singular vectors of its perturbation, with the matrix product of their bases being bounded.

Corollary 17 *[Modification of HKZ Corollary 22] Let $A \in \mathbb{R}^{m\times n}$, with $m \geq n$, have rank $k < n$, and let $U \in \mathbb{R}^{m\times k}$ be the matrix of k left singular vectors corresponding to the non-zero singular values $\sigma_1 \geq \ldots \geq \sigma_k \geq 0$ of A. Let $\widetilde{A} = A + E$. Let $\widetilde{U} \in \mathbb{R}^{m\times k}$ be the matrix of k left singular vectors corresponding to the largest k singular values $\widetilde{\sigma}_1 \geq \ldots \geq \widetilde{\sigma}_k$ of $\widetilde{A}$, and let $\widetilde{U}_\perp \in \mathbb{R}^{m\times(m-k)}$ be the remaining left singular vectors. Assume $\left\| E \right\|_2 \leq \epsilon\sigma_k$ for some $\epsilon < 1$. Then:*

1. $\widetilde{\sigma}_k \geq (1-\epsilon)\sigma_k$.

2. $\left\| \widetilde{U}_\perp^\intercal U \right\|_2 \leq \|E\|_2 / \widetilde{\sigma}_k$.

PROOF:

1. From Lemma 13,

$$\begin{aligned}
|\widetilde{\sigma}_k - \sigma_k| &\leq \|E\|_2 \\
|\widetilde{\sigma}_k - \sigma_k| &\leq \epsilon\sigma_k \\
\widetilde{\sigma}_k - \sigma_k &\geq -\epsilon\sigma_k \\
\widetilde{\sigma}_k &\geq (1-\epsilon)\sigma_k
\end{aligned}$$

which proves the first claim.

2. Recall that by Lemma 15, if Φ is a matrix of all canonical angles between $\mathrm{range}(P_{2,1})$ and $\mathrm{range}(\widehat{P}_{2,1})$, then $\sin\Phi$ contains all the singular values of $\widetilde{U}_\perp^\intercal U$ along its diagonal.

Also recall that the L_2 norm of a matrix is its top singular value. Then,

$$\begin{aligned}
\|\sin\Phi\|_2 &= \sigma_1(\sin\Phi) \quad \text{(by definition)} \\
&= \max\ \mathrm{diag}(\sin\Phi) \quad \text{(since } \sin\Phi \text{ is a diagonal matrix)} \\
&= \sigma_1(\widetilde{U}_\perp^\intercal U) \quad \text{(by Lemma 15)} \\
&= \left\| \widetilde{U}_\perp^\intercal U \right\|_2 \quad \text{(by definition)}
\end{aligned}$$

Invoking Lemma 16 with the parameter values $\delta = \widetilde{\sigma}_k$ and $\alpha = 0$ yields $\|\sin\Phi\|_2 \leq \|E\|_2 / \widetilde{\sigma}_k$. Combining this with $\|\sin\Phi\|_2 = \left\| (\widetilde{U}_\perp^\intercal U) \right\|_2$ proves claim 2.

□

A.1.3 Supporting Lemmas

In this section we develop the main supporting lemmas that help us prove Theorem 2

Estimation Errors

We define ϵ_1,$\epsilon_{2,1}$ and $\epsilon_{3,x,1}$ as sampling errors for $\vec{P}_1$,$P_{2,1}$ and $P_{3,x,1}$ respectively:

$$\epsilon_1 = \left\| \widehat{P}_1 - \vec{P}_1 \right\|_F \tag{A.6a}$$

$$\epsilon_{2,1} = \left\| \widehat{P}_{2,1} - P_{2,1} \right\|_F \tag{A.6b}$$

$$\epsilon_{3,x,1} = \left\| \widehat{P}_{3,x,1} - P_{3,x,1} \right\|_F \quad \text{for } x = 1, \ldots, n \tag{A.6c}$$

Lemma 18 *[Modification of HKZ Lemma 8] If the algorithm independently samples N observation triples from the HMM, then with probability at least $1 - \eta$:*

$$\epsilon_1 \leq \sqrt{\frac{1}{N} \ln \frac{3}{\eta}} + \sqrt{\frac{1}{N}}$$

$$\epsilon_{2,1} \leq \sqrt{\frac{1}{N} \ln \frac{3}{\eta}} + \sqrt{\frac{1}{N}}$$

$$\max_x \epsilon_{3,x,1} \leq \sqrt{\frac{1}{N} \ln \frac{3}{\eta}} + \sqrt{\frac{1}{N}}$$

$$\sum_x \epsilon_{3,x,1} \leq \min_k \left(\sqrt{\frac{k}{N} \ln \frac{3}{\eta}} + \sqrt{\frac{k}{N}} + 2\epsilon(k) \right) + \sqrt{\frac{1}{N} \ln \frac{3}{\eta}} + \sqrt{\frac{1}{N}}$$

Before proving this lemma, we need some definitions and a preliminary result. First, we restate *McDiarmid's Inequality* [108]:

Theorem 19 *Let $Z_1, \ldots, Z_m$ be independent random variables all taking values in the set $\mathcal{Z}$. Let c_i be some positive real numbers. Further, let $f : \mathcal{Z}^m \mapsto \mathbb{R}$ be a function of $Z_1, \ldots, Z_m$ that satisfies $\forall i$, $\forall z_1, \ldots, z_m, z_i' \in \mathcal{Z}$,*

$$|f(z_1, \ldots, z_i, \ldots, z_m) - f(z_1, \ldots, z_i', \ldots, z_m)| \leq c_i.$$

Then for all $\epsilon > 0$,

$$\Pr[f - \mathbb{E}[f] \geq \epsilon] \leq \exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^m c_i^2} \right).$$

Assume z is a discrete random variable that takes on values in $1, \ldots, d$. The goal is to estimate the vector $\vec{q} = [\Pr(z = j)]_{j=1}^{d}$ from N i.i.d. samples z_i $(i = 1, \ldots, N)$. Let e_j denote the j^{th} column of the $d \times d$ identity matrix. For $i = 1, \ldots, N$, suppose $\vec{q}_i$ is a column of the $d \times d$ identity matrix such that $\vec{q}_i(j) = e_{z_i}$. In other words, the z_i^{th} component of $\vec{q}_i$ is 1 and the rest are 0. Then the empirical estimate of $\vec{q}$ in terms of $\vec{q}_i$ is $\widehat{q} = \sum_{i=1}^{N} \vec{q}_i / N$.

Each part of Lemma 18 corresponds to bounding, for some $\vec{q}$, the quantity

$$\|\widehat{q} - \vec{q}\|_2^2 \quad .$$

We first state a result based on McDiarmid's inequality (Theorem 19):

Proposition 20 *[Modification of HKZ Proposition 19] For all $\epsilon > 0$ and $\widehat{q}, \vec{q}$ and N as defined above:*

$$\Pr\left(\|\widehat{q} - \vec{q}\|_2 \geq 1/\sqrt{N} + \epsilon\right) \leq e^{-N\epsilon^2}$$

PROOF: Recall $\widehat{q} = \sum_{i=1}^{N} \vec{q}_i / N$, and define $\widehat{p} = \sum_{i=1}^{N} \vec{p}_i / N$ where $\vec{p}_i = \vec{q}_i$ except for $i = k$, and p_k is an arbitrary column of the appropriate-sized identity matrix. Then we have

$$\begin{aligned}
\|\widehat{q} - \vec{q}\|_2 - \|\widehat{p} - \vec{q}\|_2 &\leq \|\widehat{q} - \widehat{p}\|_2 \quad \text{(by triangle inequality)} \\
&= \left\| (\sum_i \vec{q}_i)/N - (\sum_i \vec{p}_i)/N \right\|_2 \\
&= (1/N)\,\|\vec{q}_k - \vec{p}_k\|_2 \quad \text{(by definition of } \widehat{p}, \widehat{q} \text{ and } L_2\text{-norm)} \\
&\leq (1/N)\sqrt{1^2 + 1^2} \\
&= \sqrt{2}/N
\end{aligned}$$

This shows that $\|\widehat{q} - \vec{q}\|_2$ is a function of random variables $\vec{q}_1, \ldots, \vec{q}_N$ such changing the k^{th} random variable q_k for any $1 \leq k \leq N$ (resulting in $\|\widehat{p} - \vec{q}\|_2$) changes the value of the function by at most $c_k = \sqrt{2}/N$. Note that $\vec{q}$ is not a random variable but rather the variable we are trying to estimate. In this case, McDiarmid's inequality (Theorem 19) bounds the deviation $\|\widehat{q} - \vec{q}\|_2$ from its expectation $\mathbb{E}\,\|\widehat{q} - \vec{q}\|_2$ as:

$$\begin{aligned}
\Pr(\|\widehat{q} - \vec{q}\|_2 \geq \mathbb{E}\,\|\widehat{q} - \vec{q}\|_2 + \epsilon) &\leq \exp \frac{-2\epsilon^2}{\sum_{i=1}^N c_i^2} \\
&= \exp \frac{-2\epsilon^2}{N \cdot 2/N^2} \\
&= e^{-N\epsilon^2} \qquad \text{(A.7)}
\end{aligned}$$

We can bound the expected value using the following inequality:

$$\begin{aligned}
\mathbb{E}\left\|\sum_{i=1}^N \vec{q}_i - N\vec{q}\right\|_2 &= \mathbb{E}\left(\left\|\sum_{i=1}^N \vec{q}_i - N\vec{q}\right\|_2^2\right)^{1/2} \\
&\leq \left(\mathbb{E}\left\|\sum_{i=1}^N \vec{q}_i - N\vec{q}\right\|_2^2\right)^{1/2} \quad \text{(by concavity of square root, and Jensens inequality)} \\
&= \left(\sum_{i=1}^N \mathbb{E}\,\|\vec{q}_i - \vec{q}\|_2^2\right)^{1/2} \\
&= \left(\sum_{i=1}^N \mathbb{E}(\vec{q}_i - \vec{q})^\mathsf{T}(\vec{q}_i - \vec{q})\right)^{1/2}
\end{aligned}$$

Multiplying out and using linearity of expectation and properties of $\vec{q}_i$ (namely, that $\vec{q}_i^\mathsf{T}\vec{q}_i = 1$, $\mathbb{E}(\vec{q}_i) = \vec{q}$ and $\vec{q}$ is constant), we get:

$$\begin{aligned}
\mathbb{E}\left\|\sum_{i=1}^N \vec{q}_i - N\vec{q}\right\|_2 &\leq \left(\sum_{i=1}^N \mathbb{E}(1 - 2\vec{q}_i^\mathsf{T}\vec{q} + \|\vec{q}\|_2^2)\right)^{1/2} \quad \text{(since } \vec{q}_i^\mathsf{T}\vec{q}_i = 1\text{)} \\
&= \left(\sum_{i=1}^N \mathbb{E}(1) - 2\sum_{i=1}^N \mathbb{E}(\vec{q}_i^\mathsf{T}\vec{q}) + \sum_{i=1}^N \mathbb{E}\,\|\vec{q}\|_2^2\right)^{1/2} \\
&= \left(N - 2N\,\|\vec{q}\|_2^2 + N\,\|\vec{q}\|_2^2\right)^{1/2} \\
&= \sqrt{N(1 - \|\vec{q}\|_2^2)}
\end{aligned}$$

This implies an upper bound on the expected value:

$$\begin{aligned}
\mathbb{E}\left\|\widehat{q}-\vec{q}\right\|_2^2 &= (1/N^2)\mathbb{E}\left\|\sum_{i=1}^{N}\vec{q}_i - N\vec{q}\right\|_2^2 \\
&\leq (1/N^2)\cdot N(1-\|\vec{q}\|_2^2) \\
\Rightarrow \mathbb{E}\left\|\widehat{q}-\vec{q}\right\|_2 &\leq (1/\sqrt{N})\sqrt{(1-\|\vec{q}\|_2^2)} \\
&\leq (1/\sqrt{N})
\end{aligned}$$

Using this upper bound in McDiarmids inequality (equation (A.7)), we get a looser version of the bound that proves the proposition:

$$\Pr(\|\widehat{q}-\vec{q}\|_2 \geq 1/\sqrt{N}+\epsilon) \leq e^{-N\epsilon^2}$$

□

We are now ready to prove Lemma 18.

PROOF:[Lemma 18] We will treat $\widehat{P}_1$,$\widehat{P}_{2,1}$ and $\widehat{P}_{3,x,1}$ as vectors, and use McDiarmid's inequality to bound the error in estimating a distribution over a simplex based on indicator vector samples, using Proposition 20. We know that

$$\Pr\left(\|\widehat{q}-\vec{q}\|_2 \geq \sqrt{1/N}+\epsilon\right) \leq e^{-N\epsilon^2} \quad .$$

Now let $\eta = e^{-N\epsilon^2}$. This implies

$$\begin{aligned}
\ln\eta &= -N\epsilon^2 \\
\ln(1/\eta) &= N\epsilon^2 \\
\epsilon &= \sqrt{\ln(1/\eta)/N}
\end{aligned}$$

Hence,

$$\Pr\left(\|\widehat{q}-\vec{q}\|_2 \geq \sqrt{1/N}+\sqrt{\ln(1/\eta)/N}\right) \leq \eta$$

Therefore, with probability at least $1-\eta$,

$$\|\widehat{q}-\vec{q}\|_2 \leq 1/\sqrt{N}+\sqrt{\ln(1/\eta)/N} \qquad \text{(A.8)}$$

Now, in place of $\vec{q}$ in equation (A.8), we substitute the stochastic vector $\vec{P}_1$ to prove the first claim, the vectorized version of the stochastic matrix $P_{2,1}$ to prove the second claim, and the vectorized version of the stochastic *tensor* $P_{3,2,1} \in \mathbb{R}^{n\times n\times n}$ obtained by stacking $P_{3,x,1}$ matrices over all x, to prove the third claim. The matrices $\widehat{P}_{3,x,1}$ are stacked accordingly to obtain the estimated tensor $\widehat{P}_{3,2,1}$. We get the following:

$$\begin{aligned}
\epsilon_1 &\leq 1/\sqrt{N} + \sqrt{\ln(1/\eta)/N} \quad \text{(hence proving the first claim)} \\
\epsilon_{2,1} &\leq 1/\sqrt{N} + \sqrt{\ln(1/\eta)/N} \quad \text{(hence proving the second claim)} \\
\max_x \epsilon_{3,x,1} &\leq \sqrt{\sum_x {\epsilon_{3,x,1}}^2} \\
&= \sqrt{\sum_x \left\| P_{3,x,1} - \widehat{P}_{3,x,1} \right\|_2^2} \\
&= \sqrt{\sum_x \sum_i \sum_j ([P_{3,x,1}]_{i,j} - [\widehat{P}_{3,x,1}]_{i,j})^2} \\
&= \sqrt{\left\| P_{3,2,1} - \widehat{P}_{3,2,1} \right\|_2^2} \\
&= \left\| P_{3,2,1} - \widehat{P}_{3,2,1} \right\|_2 \\
&\leq \sqrt{1/N} + \sqrt{\ln(1/\eta)/N} \quad \text{(hence proving the third claim)}
\end{aligned}$$

Note the following useful inequality from the above proof:

$$\sqrt{\sum_x {\epsilon_{3,x,1}}^2} \leq \sqrt{1/N} + \sqrt{\ln(1/\eta)/N} \tag{A.9}$$

It remains to prove the fourth claim, regarding $\sum_x \epsilon_{3,x,1}$. First we get a bound that depends on n as follows:

$$\begin{aligned}
\sum_x \epsilon_{3,x,1} &= \sum_x |\epsilon_{3,x,1}| \quad (\because \forall x, \epsilon_{3,x,1} \geq 0) \\
&\leq \sqrt{n}\sqrt{\sum_x {\epsilon_{3,x,1}}^2} \quad (\because \forall \vec{x} \in \mathbb{R}^a, \|\vec{x}\|_1 \leq \sqrt{a}\, \|\vec{x}\|_2) \\
&\leq \sqrt{n/N} + \sqrt{\frac{n}{N} \ln \frac{1}{\eta}}
\end{aligned}$$

We aren't going to use the above bound. Instead, if n is large and N small, this bound can be improved by removing direct dependence on n. Let $\epsilon(k)$ be the sum of smallest $n-k$ probabilities of the second observation x_2. Let S_k be the set of these $n-k$ such observations x, for any k. Therefore,

$$\epsilon(k) = \sum_{x \in S_k} \Pr[x_2 = x] = \sum_{x \in S_k} \sum_{i,j} [P_{3,x,1}]_{ij}$$

Now, first note that we can bound $\sum_{x \notin S_k} \epsilon_{3,x,1}$ as follows:

$$\begin{aligned} \sum_{x \notin S_k} \epsilon_{3,x,1} &\leq \sum_{x \notin S_k} |\epsilon_{3,x,1}| \\ &\leq \sqrt{k} \sqrt{\sum_{x \notin S_k} {\epsilon_{3,x,1}}^2} \quad (\because \forall \vec{x} \in \mathbb{R}^a, \|\vec{x}\|_1 \leq \sqrt{a}\,\|\vec{x}\|_2) \end{aligned}$$

By combining with equation A.9, we get

$$\sum_{x \notin S_k} \epsilon_{3,x,1} \leq \sqrt{k/N} + \sqrt{k \ln(1/\eta)/N} \tag{A.10}$$

To bound $\sum_{x \in S_k} \epsilon_{3,x,1}$, we first apply equation (A.8) again. Consider the vector $\vec{q}$ of length $kn^2 + 1$ whose first kn^2 entries comprise the elements of $P_{3,x,1}$ for all $x \notin S_k$, and whose last entry is the cumulative sum of elements of $P_{3,x,1}$ for all $x \in S_k$. Define $\widehat{q}$ accordingly with $\widehat{P}_{3,x,1}$ instead of $P_{3,x,1}$. Now equation (A.8) directly gives us with probability at least $1-\eta$:

$$\left[\sum_{x \notin S_k} \sum_{i,j} ([\widehat{P}_{3,x,1}]_{i,j} - [P_{3,x,1}]_{i,j})^2 + \left| \sum_{x \in S_k} \sum_{i,j} ([\widehat{P}_{3,x,1}]_{ij} - \sum_{x \in S_k} \sum_{i,j} [P_{3,x,1}]_{ij}) \right|^2 \right]^{\frac{1}{2}} \leq \sqrt{1/N} + \sqrt{\ln(1/\eta)/N}$$

$$\sum_{x \notin S_k} \left\| \widehat{P}_{3,x,1} - P_{3,x,1} \right\|_F^2 + \left| \sum_{x \in S_k} \sum_{i,j} ([\widehat{P}_{3,x,1}]_{ij} - [P_{3,x,1}]_{ij}) \right|^2 \leq \left(\sqrt{1/N} + \sqrt{\ln(1/\eta)/N} \right)^2$$

Since the first term above is positive, we get

$$\left| \sum_{x \in S_k} \sum_{i,j} ([\widehat{P}_{3,x,1}]_{ij} - [P_{3,x,1}]_{ij}) \right| \leq \sqrt{1/N} + \sqrt{\ln(1/\eta)/N} \tag{A.11}$$

Now, by definition of S_k:,

$$\begin{aligned}
\sum_{x\in S_k} \epsilon_{3,x,1} &= \sum_{x\in S_k} \left\| \widehat{P}_{3,x,1} - P_{3,x,1} \right\|_F \\
&\leq \sum_{x\in S_k} \sum_{i,j} \left| [\widehat{P}_{3,x,1}]_{ij} - [P_{3,x,1}]_{ij} \right| \quad (\because \forall \vec{x}, \|\vec{x}\|_2 \leq \|\vec{x}\|_1) \\
&= \sum_{x\in S_k} \sum_{i,j} \max\left(0, [\widehat{P}_{3,x,1}]_{ij} - [P_{3,x,1}]_{ij}\right) \\
&\quad - \sum_{x\in S_k} \sum_{i,j} \min\left(0, [\widehat{P}_{3,x,1}]_{ij} - [P_{3,x,1}]_{ij}\right) \quad (\because \forall \vec{x}, |\vec{x}| = [\max(0,\vec{x}) - \min(0,\vec{x})]) \\
&\leq \sum_{x\in S_k} \sum_{i,j} \max\left(0, [\widehat{P}_{3,x,1}]_{ij} - [P_{3,x,1}]_{ij}\right) + \sum_{x\in S_k} \sum_{i,j} [P_{3,x,1}]_{ij} \\
&\quad + \sum_{x\in S_k} \sum_{i,j} \min\left(0, [\widehat{P}_{3,x,1}]_{ij} - [P_{3,x,1}]_{ij}\right) + \sum_{x\in S_k} \sum_{i,j} [P_{3,x,1}]_{ij} \\
&= \sum_{x\in S_k} \sum_{i,j} \max\left(0, [\widehat{P}_{3,x,1}]_{ij} - [P_{3,x,1}]_{ij}\right) + \epsilon(k) \\
&\quad + \sum_{x\in S_k} \sum_{i,j} \min\left(0, [\widehat{P}_{3,x,1}]_{ij} - [P_{3,x,1}]_{ij}\right) + \epsilon(k) \quad (\text{by definition of } \epsilon(k)) \\
&\leq \left| \sum_{x\in S_k} \sum_{i,j} \left([\widehat{P}_{3,x,1}]_{ij} - [P_{3,x,1}]_{ij} \right) \right| + 2\epsilon(k)
\end{aligned}$$

Plugging in equation (A.11), we get a bound on $\sum_{x\in S_k} \epsilon_{3,x,1}$:

$$\sum_{x\in S_k} \epsilon_{3,x,1} \leq \sqrt{1/N} + \sqrt{\ln(1/\eta)/N} + 2\epsilon(k)$$

Combining with equation (A.10) and noting that k is arbitrary, we get the desired bound:

$$\sum_{x} \epsilon_{3,x,1} \leq \min_k [\sqrt{k \ln(1/\eta)/N} + \sqrt{k/N} + \sqrt{\ln(1/\eta)/N} + \sqrt{1/N} + 2\epsilon(k)]$$

Note that, to get the term $\ln(3/\eta)$ instead of $\ln(1/\eta)$ as in the fourth claim, we simply use $\eta/3$ instead of η. This bound on $\sum_x \epsilon_{3,x,1}$ will be small if the number of frequently occurring observations is small, even if n itself is large. □

The next lemma uses the perturbation bound in Corollary 17 to bound the effect of sampling error on the estimate $\widehat{U}$, and on the conditioning of $(\widehat{U}^\top OR)$.

Lemma 21 *[Modification of HKZ Lemma 9] Suppose* $\epsilon_{2,1} \leq \varepsilon \cdot \sigma_k(P_{2,1})$ *for some* $\varepsilon < 1/2$. *Let* $\varepsilon_0 = \epsilon_{2,1}^2/((1-\varepsilon)\sigma_k(P_{2,1}))^2$. *Define* $U, \widehat{U} \in \mathbb{R}^{m \times k}$ *as the matrices of the first* k *left singular vectors of* $P_{2,1}, \widehat{P}_{2,1}$ *respectively. Let* $\theta_1, \ldots, \theta_k$ *be the canonical angles between* $\mathrm{span}(U)$ *and* $\mathrm{span}(\widehat{U})$. *Then:*

1. $\varepsilon_0 < 1$

2. $\sigma_k(\widehat{U}^\top \widehat{P}_{2,1}) \geq (1-\varepsilon)\sigma_k(P_{2,1})$

3. $\sigma_k(\widehat{U}^\top P_{2,1}) \geq \sqrt{1-\varepsilon_0}\,\sigma_k(P_{2,1})$

4. $\sigma_k(\widehat{U}^\top OR) \geq \sqrt{1-\varepsilon_0}\,\sigma_k(OR)$

PROOF: First some additional definitions and notation. Define $\widehat{U}_\perp$ to be the remaining $n-k$ left singular vectors of $\widehat{P}_{2,1}$ corresponding to the lower $n-k$ singular values, and correspondingly $U_\perp$ for $P_{2,1}$. Suppose $U\Sigma V^\top = P_{2,1}$ is the thin SVD of $P_{2,1}$. Finally, we use the notation $\vec{\nu}_i\{A\} \in \mathbb{R}^q$ to denote the i^{th} *right singular vector* of a matrix $A \in \mathbb{R}^{p \times q}$. Recall that $\sigma_i(A) = \|A\vec{\nu}_i\{A\}\|_2$ by definition.

First claim: $\varepsilon_0 < 1$ follows from the assumptions:

$$\begin{aligned}\varepsilon_0 &= \frac{{\epsilon_{2,1}}^2}{((1-\varepsilon)\sigma_k(P_{2,1}))^2} \\ &\leq \frac{\varepsilon^2 \sigma_k(P_{2,1})^2}{(1-\varepsilon)^2 \sigma_k(P_{2,1})^2} \\ &= \frac{\varepsilon^2}{(1-\varepsilon)^2} \\ &< 1 \quad \text{(since } \varepsilon < 1/2\text{)}\end{aligned}$$

Second claim: By Corollary 17, $\sigma_k(\widehat{P}_{2,1}) \geq (1-\varepsilon)\sigma_k(P_{2,1})$. The second claim follows from noting that $\sigma_k(\widehat{U}^\top \widehat{P}_{2,1}) = \sigma_k(\widehat{P}_{2,1})$.

Third and fourth claims: First consider the k^{th} singular value of $\widehat{U}^\mathsf{T} U$. For any vector $x \in \mathbb{R}^k$:

$$\begin{aligned}
\frac{\left\|\widehat{U}^\mathsf{T} U x\right\|_2}{\|x\|_2} &\geq \min_y \frac{\left\|\widehat{U}^\mathsf{T} U y\right\|_2}{\|y\|_2} \\
&= \sigma_k(\widehat{U}^\mathsf{T} U) \quad \text{(by definition of smallest singular value)} \\
&= \cos(\theta_k) \quad \text{(by Definition 14)} \\
&= \sqrt{1 - \sin^2(\theta_k)} \\
&= \sqrt{1 - \sigma_k(\widehat{U}_\perp^\mathsf{T} U)^2} \quad \text{(by Lemma 15)} \\
&\geq \sqrt{1 - \sigma_1(\widehat{U}_\perp^\mathsf{T} U)^2} \\
&= \sqrt{1 - \left\|\widehat{U}_\perp^\mathsf{T} U\right\|_2^2} \quad \text{(by definition of } L_2 \text{ matrix norm)}
\end{aligned}$$

Therefore,

$$\left\|\widehat{U}^\mathsf{T} U x\right\|_2 \geq \|x\|_2 \sqrt{1 - \left\|\widehat{U}_\perp^\mathsf{T} U\right\|_2^2} \tag{A.12}$$

Note that

$$\begin{aligned}
\left\|\widehat{U}_\perp^\mathsf{T} U\right\|_2^2 &\leq {\epsilon_{2,1}}^2 / \sigma_k(P_{2,1})^2 \quad \text{(by Corollary 17)} \\
&\leq \frac{{\epsilon_{2,1}}^2}{(1-\varepsilon)^2 \sigma_k(P_{2,1})^2} \quad \text{(since } 0 \leq \varepsilon < 1/2\text{)} \\
&= \varepsilon_0 \quad \text{(by definition)}
\end{aligned}$$

Hence, by combining the above with equation (A.12), since $0 \leq \varepsilon_0 < 1$:

$$\left\|\widehat{U}^\mathsf{T} U x\right\|_2 \geq \|x\|_2 \sqrt{1 - \varepsilon_0} \quad \text{(for all } x \in \mathbb{R}^k\text{)} \tag{A.13}$$

The remaining claims follow by taking different choices of x in equation (A.13), and by using the intuition that the smallest singular value of a matrix is the smallest possible L_2 norm of a unit-length vector after the matrix has left-multiplied that vector, and the particular vector for which this holds is the corresponding right singular vector. For claim 3, let $x = \Sigma V^\top \vec{\nu}_k\{\widehat{U}^\top P_{2,1}\}$. Then by equation (A.13):

$$\left\|\widehat{U}^\top U\Sigma V^\top \vec{\nu}_k\{\widehat{U}^\top P_{2,1}\}\right\|_2 \geq \left\|\Sigma V^\top \vec{\nu}_k\{\widehat{U}^\top P_{2,1}\}\right\|_2 \sqrt{1-\varepsilon_0}$$

Since $P_{2,1} = U\Sigma V^\top$, and $\left\|\Sigma V^\top \vec{\nu}_k\{\Sigma V^\top\}\right\|_2 \leq \left\|\Sigma V^\top \vec{\nu}_k\{\widehat{U}^\top P_{2,1}\}\right\|_2$ by definition of $\vec{\nu}_k\{\Sigma V^\top\}$, we have:

$$\begin{aligned}
\left\|\widehat{U}^\top P_{2,1}\vec{\nu}_k\{\widehat{U}^\top P_{2,1}\}\right\|_2 &\geq \left\|\Sigma V^\top \vec{\nu}_k\{\Sigma V^\top\}\right\|_2 \sqrt{1-\varepsilon_0} \\
\sigma_k(\widehat{U}^\top P_{2,1}) &\geq \sigma_k(\Sigma V^\top)\sqrt{1-\varepsilon_0} \quad \text{(by definition of } \sigma_k(\widehat{U}^\top P_{2,1}), \sigma_k(\Sigma V^\top)) \\
\sigma_k(\widehat{U}^\top P_{2,1}) &\geq \sigma_k(P_{2,1})\sqrt{1-\varepsilon_0} \quad (\because \sigma_k(\Sigma V^\top) = \sigma_k(P_{2,1}))
\end{aligned}$$

which proves claim 3.

For claim 4, first recall that OR can be exactly expressed as $P_{2,1}(S\ \mathrm{diag}(\vec{\pi})O^\top)^+$ (equation (A.2)). For brevity, let $\mathcal{A} = (S\ \mathrm{diag}(\vec{\pi})O^\top)^+$, so that $OR = P_{2,1}\mathcal{A}$. Then, let $x = \Sigma V^\top \mathcal{A}\vec{\nu}_k\{\widehat{U}^\top OR\}$ in equation (A.13):

$$\left\|\widehat{U}^\top U\Sigma V^\top \mathcal{A}\vec{\nu}_k\{\widehat{U}^\top OR\}\right\|_2 \geq \left\|\Sigma V^\top \mathcal{A}\vec{\nu}_k\{\widehat{U}^\top OR\}\right\|_2 \sqrt{1-\varepsilon_0}$$

Since $P_{2,1} = U\Sigma V^\top$, and $\left\|\Sigma V^\top \mathcal{A}\vec{\nu}_k\{\Sigma V^\top \mathcal{A}\}\right\|_2 \leq \left\|\Sigma V^\top \mathcal{A}\vec{\nu}_k\{\widehat{U}^\top OR\}\right\|_2$ by definition of $\vec{\nu}_k\{\Sigma V^\top \mathcal{A}\}$, we get:

$$\begin{aligned}
\left\|\widehat{U}^\top P_{2,1}\mathcal{A}\vec{\nu}_k\{\widehat{U}^\top OR\}\right\|_2 &\geq \left\|\Sigma V^\top \mathcal{A}\vec{\nu}_k\{\Sigma V^\top \mathcal{A}\}\right\|_2 \sqrt{1-\varepsilon_0} \\
\left\|\widehat{U}^\top OR\vec{\nu}_k\{\widehat{U}^\top OR\}\right\|_2 &\geq \sigma_k(\Sigma V^\top \mathcal{A})\sqrt{1-\varepsilon_0} \quad \text{(by equation (A.2))}
\end{aligned}$$

By definition of $\sigma_k(\widehat{U}^\top OR), \sigma_k(\Sigma V^\top \mathcal{A})$, we see that

$$\sigma_k(\widehat{U}^\top OR) \geq \sigma_k(\Sigma V^\top \mathcal{A})\sqrt{1-\varepsilon_0}$$
$$\sigma_k(\widehat{U}^\top OR) \geq \sigma_k(OR)\sqrt{1-\varepsilon_0} \quad (\because \sigma_k(\Sigma V^\top \mathcal{A}) = \sigma_k(P_{2,1}\mathcal{A}) = \sigma_k(OR))$$

hence proving claim 4. □

Define the following observable representation using $U = \widehat{U}$, which constitutes a *true* observable representation for the HMM as long as $(U^\top OR)$ is invertible:

$$\widetilde{b}_\infty = (P_{2,1}{}^\top \widehat{U})^+ \vec{P}_1 = (\widehat{U}^\top OR)^{-T} R^\top \vec{1}_m$$
$$\widetilde{B}_x = (\widehat{U}^\top P_{3,x,1})(\widehat{U}^\top P_{2,1})^+ = (\widehat{U}^\top OR) W_x (\widehat{U}^\top OR)^{-1} \quad \text{for } x = 1, \ldots, n$$
$$\widetilde{b}_1 = \widehat{U}^\top \vec{P}_1$$

Define the following error measures of estimated parameters with respect to the true observable representation. The error vector in δ_1 is projected to $\mathbb{R}^m$ before applying the vector norm, for convenience in later theorems.

$$\delta_\infty = \left\| (\widehat{U}^\top O)^\top (\widehat{b}_\infty - \widetilde{b}_\infty) \right\|_\infty$$
$$\Delta_x = \left\| (\widehat{U}^\top OR)^{-1} \left(\widehat{B}_x - \widetilde{B}_x \right) (\widehat{U}^\top OR) \right\|_1 = \left\| (\widehat{U}^\top OR)^{-1} \widehat{B}_x (\widehat{U}^\top OR) - W_x \right\|_1$$
$$\Delta = \sum_x \Delta_x$$
$$\delta_1 = \left\| R(\widehat{U}^\top OR)^{-1} (\widehat{b}_1 - \widetilde{b}_1) \right\|_1 = \left\| R(\widehat{U}^\top OR)^{-1} \widehat{b}_1 - \vec{\pi} \right\|_1$$

The next Lemma proves that the estimated parameters $\widehat{b}_\infty, \widehat{B}_x, \widehat{b}_1$ are close to the true parameters $\widetilde{b}_\infty, \widetilde{B}_x, \widetilde{b}_1$ if the sampling errors $\epsilon_1, \epsilon_{2,1}, \epsilon_{3,x,1}$ are small:

Lemma 22 *[Modification of HKZ Lemma 10] Assume* $\epsilon_{2,1} < \sigma_k(P_{2,1})/3$. *Then:*

$$\delta_\infty \leq 4 \cdot \left(\frac{\epsilon_{2,1}}{\sigma_k(P_{2,1})^2} + \frac{\epsilon_1}{3\sigma_k(P_{2,1})} \right)$$

$$\Delta_x \leq \frac{8}{\sqrt{3}} \cdot \frac{\sqrt{k}}{\sigma_k(OR)} \cdot \left(\Pr[x_2 = x] \cdot \frac{\epsilon_{2,1}}{\sigma_k(P_{2,1})^2} + \frac{\Sigma_x \epsilon_{3,x,1}}{3\sigma_k(P_{2,1})} \right)$$

$$\Delta \leq \frac{8}{\sqrt{3}} \cdot \frac{\sqrt{k}}{\sigma_k(OR)} \cdot \left(\frac{\epsilon_{2,1}}{\sigma_k(P_{2,1})^2} + \frac{\Sigma_x \epsilon_{3,x,1}}{3\sigma_k(P_{2,1})} \right)$$

$$\delta_1 \leq \frac{2}{\sqrt{3}} \cdot \frac{\sqrt{k}}{\sigma_k(OR)} \cdot \epsilon_1$$

PROOF: Note that the assumption on $\epsilon_{2,1}$ guarantees $(\widehat{U}^\mathsf{T} OR)$ to be invertible by Lemma 21, claim 4.

δ_∞ bound: We first see that δ_∞ can be bounded by $\left\| \widehat{b}_\infty - \widetilde{b}_\infty \right\|_2$:

$$\begin{aligned}
\delta_\infty &= \left\| (O^\mathsf{T} U)(\widehat{b}_\infty - \widetilde{b}_\infty) \right\|_\infty \\
&\leq \left\| O^\mathsf{T} \right\|_\infty \left\| U(\widehat{b}_\infty - \widetilde{b}_\infty) \right\|_\infty \\
&\leq \left\| U(\widehat{b}_\infty - \widetilde{b}_\infty) \right\|_\infty \\
&\leq \left\| U(\widehat{b}_\infty - \widetilde{b}_\infty) \right\|_2 \\
&\leq \left\| \widehat{b}_\infty - \widetilde{b}_\infty \right\|_2
\end{aligned}$$

In turn, this leads to the following expression:

$$\begin{aligned}
\left\|\widehat{b}_\infty - \widetilde{b}_\infty\right\|_2 &= \left\|(\widehat{P}_{2,1}^{\mathsf{T}}\widehat{U})^+\widehat{P}_1 - (P_{2,1}{}^{\mathsf{T}}\widehat{U})^+\vec{P}_1\right\|_2 \\
&= \left\|(\widehat{P}_{2,1}^{\mathsf{T}}\widehat{U})^+\widehat{P}_1 - (P_{2,1}{}^{\mathsf{T}}\widehat{U})^+\widehat{P}_1 + (P_{2,1}{}^{\mathsf{T}}\widehat{U})^+\widehat{P}_1 - (P_{2,1}{}^{\mathsf{T}}\widehat{U})^+\vec{P}_1\right\|_2 \\
&= \left\|\left((\widehat{P}_{2,1}^{\mathsf{T}}\widehat{U})^+ - (P_{2,1}{}^{\mathsf{T}}\widehat{U})^+\right)\widehat{P}_1 + (P_{2,1}{}^{\mathsf{T}}\widehat{U})^+(\widehat{P}_1 - \vec{P}_1)\right\|_2 \\
&\leq \left\|((\widehat{P}_{2,1}^{\mathsf{T}}\widehat{U})^+ - (P_{2,1}{}^{\mathsf{T}}\widehat{U})^+)\widehat{P}_1\right\|_2 + \left\|(P_{2,1}{}^{\mathsf{T}}\widehat{U})^+(\widehat{P}_1 - \vec{P}_1)\right\|_2 \\
&\leq \left\|((\widehat{P}_{2,1}^{\mathsf{T}}\widehat{U})^+ - (P_{2,1}{}^{\mathsf{T}}\widehat{U})^+)\right\|_2 \left\|\widehat{P}_1\right\|_1 + \left\|(P_{2,1}{}^{\mathsf{T}}\widehat{U})^+\right\|_2 \left\|(\widehat{P}_1 - \vec{P}_1)\right\|_2
\end{aligned}$$

The last step above obtains from the consistency of the L_2 matrix norm with L_1 vector norm, and from the definition of L_2 matrix norm (spectral norm) as $\|A\|_2 = \max \frac{\|Ax\|_2}{\|x\|_2}$. Now, recall that $\widehat{U}$ has orthonormal columns, and hence multiplying a matrix with $\widehat{U}$ cannot increase its spectral norm. Hence,

$$\left\|\widehat{P}_{2,1}^{\mathsf{T}}\widehat{U} - P_{2,1}{}^{\mathsf{T}}\widehat{U}\right\|_2 = \left\|(\widehat{P}_{2,1}^{\mathsf{T}} - P_{2,1}{}^{\mathsf{T}})\widehat{U}\right\|_2 \leq \left\|\widehat{P}_{2,1}^{\mathsf{T}} - P_{2,1}{}^{\mathsf{T}}\right\|_2 = \epsilon_{2,1}.$$

So, we can use Lemma 12 to bound the L_2-distance between pseudoinverses of $\widehat{P}_{2,1}^{\mathsf{T}}\widehat{U}$ and $P_{2,1}{}^{\mathsf{T}}\widehat{U}$ using $\epsilon_{2,1}$ as an upper bound on the difference between the matrices themselves. Also recall that singular values of the pseudoinverse of a matrix are the reciprocals of the matrix singular values. Substituting this in the above expression, along with the facts that $\sigma_k(\widehat{P}_{2,1}^{\mathsf{T}}\widehat{U}) = \sigma_k(\widehat{P}_{2,1})$, $\left\|\widehat{P}_1\right\|_1 = 1$ and $\left\|(\widehat{P}_1 - \vec{P}_1)\right\|_2 = \epsilon_1$, gives us:

$$\left\|\widehat{b}_\infty - \widetilde{b}_\infty\right\|_2 \leq \frac{1+\sqrt{5}}{2} \cdot \frac{\epsilon_{2,1}}{\min\left(\sigma_k(\widehat{P}_{2,1}), \sigma_k(P_{2,1}{}^{\mathsf{T}}\widehat{U})\right)^2} + \frac{\epsilon_1}{\sigma_k(P_{2,1}{}^{\mathsf{T}}\widehat{U})}$$

Now, to simplify the last expression further, consider Lemma 21 in the above context. Here, $\epsilon_{2,1} \leq \sigma_k(P_{2,1})/3$ and hence $\varepsilon = 1/3$. Therefore $\sigma_k(\widehat{P}_{2,1}) = \sigma_k(\widehat{U}^{\mathsf{T}}\widehat{P}_{2,1}) \geq (2/3)\sigma_k(P_{2,1})$ and $\sigma_k(\widehat{U}^{\mathsf{T}}P_{2,1}) \geq \sqrt{1-\varepsilon_0}\sigma_k(P_{2,1})$. Hence

$$\min\left(\sigma_k(\widehat{P}_{2,1}), \sigma_k(P_{2,1}{}^{\mathsf{T}}\widehat{U})\right)^2 = \sigma_k(P_{2,1})^2 \cdot \min(2/3, \sqrt{1-\varepsilon_0})^2$$

The latter term is larger since

$$\begin{aligned}\varepsilon_0 &= \frac{{\epsilon_{2,1}}^2}{((1-\varepsilon)\sigma_k(P_{2,1}))^2}\\ &\leq \frac{\sigma_k(P_{2,1})^2/9}{4\sigma_k(P_{2,1})^2/9}\\ &= 1/4\\ \Rightarrow \sqrt{1-\varepsilon_0} &\geq \sqrt{3}/2 > 2/3\end{aligned}$$

Therefore $\min\left(\sigma_k(\widehat{P}_{2,1}), \sigma_k({P_{2,1}}^\mathsf{T}\widehat{U})\right)^2 \geq \sigma_k(P_{2,1})^2(2/3)^2$. Plugging this into the expression above along with the fact that $\sigma_k(\widehat{U}^\mathsf{T}P_{2,1}) \geq (\sqrt{3}/2)\sigma_k(P_{2,1})$, we prove the required result for δ_∞:

$$\begin{aligned}\delta_\infty &\leq \frac{1+\sqrt{5}}{2}\cdot\frac{9\epsilon_{2,1}}{4\sigma_k(P_{2,1})^2} + \frac{2\epsilon_1}{\sqrt{3}\sigma_k(P_{2,1})}\\ &\leq 4\cdot\left(\frac{\epsilon_{2,1}}{\sigma_k(P_{2,1})^2} + \frac{\epsilon_1}{\sigma_k(P_{2,1})}\right)\end{aligned}$$

Δ_x,Δ bounds: . We first bound each term Δ_x by $\sqrt{k}\left\|\widehat{B}_x - \widetilde{B}_x\right\|_2/\sigma_k((\widehat{U}^\mathsf{T}OR))$:

$$\begin{aligned}\Delta_x &= \left\|(\widehat{U}^\mathsf{T}OR)^{-1}\left(\widehat{B}_x - \widetilde{B}_x\right)(\widehat{U}^\mathsf{T}OR)\right\|_1\\ &\leq \left\|(\widehat{U}^\mathsf{T}OR)^{-1}(\widehat{B}_x - \widetilde{B}_x)\widehat{U}^\mathsf{T}\right\|_1\|OR\|_1 \quad \text{(by norm consistency)}\\ &\leq \sqrt{k}\left\|(\widehat{U}^\mathsf{T}OR)^{-1}(\widehat{B}_x - \widetilde{B}_x)\widehat{U}^\mathsf{T}\right\|_2\|OR\|_1 \quad \text{(by } L_1 \text{ vs. } L_2 \text{ norm inequality)}\\ &\leq \sqrt{k}\left\|(\widehat{U}^\mathsf{T}OR)^{-1}\right\|_2\left\|\widehat{B}_x - \widetilde{B}_x\right\|_2\left\|\widehat{U}^\mathsf{T}\right\|_2\|O\|_1\|R\|_1 \quad \text{(by norm consistency)}\\ &\leq \sqrt{k}\left\|(\widehat{U}^\mathsf{T}OR)^{-1}\right\|_2\left\|\widehat{B}_x - \widetilde{B}_x\right\|_2 \quad \left(\left\|\widehat{U}^\mathsf{T}\right\|_2, \|O\|_1, \|R\|_1 \leq 1\right)\\ &= \sqrt{k}\left\|\widehat{B}_x - \widetilde{B}_x\right\|_2/\sigma_k(\widehat{U}^\mathsf{T}OR) \quad (\because \sigma_{\max}(\widehat{U}^\mathsf{T}OR)^{-1} = 1/\sigma_{\min}(\widehat{U}^\mathsf{T}OR))\end{aligned}$$

The term $\left\|\widehat{B}_x - \widetilde{B}_x\right\|_2$ in the numerator can be bounded by

$$\begin{aligned}
\left\|\widehat{B}_x - \widetilde{B}_x\right\|_2 &= \left\|(\widehat{U}^\top P_{3,x,1})(\widehat{U}^\top P_{2,1})^+ - (\widehat{U}^\top \widehat{P}_{3,x,1})(\widehat{U}^\top \widehat{P}_{2,1})^+\right\|_2 \\
&\leq \left\|(\widehat{U}^\top P_{3,x,1})\left((\widehat{U}^\top P_{2,1})^+ - (\widehat{U}^\top \widehat{P}_{2,1})^+\right)\right\|_2 + \left\|\widehat{U}^\top\left(P_{3,x,1} - \widehat{P}_{3,x,1}\right)(\widehat{U}^\top P_{2,1})^+\right\|_2 \\
&\leq \|P_{3,x,1}\|_2 \cdot \frac{1+\sqrt{5}}{2} \cdot \frac{\epsilon_{2,1}}{\min\left(\sigma_k(\widehat{P}_{2,1}), \sigma_k(\widehat{U}^\top P_{2,1})\right)^2} + \frac{\epsilon_{3,x,1}}{\sigma_k(\widehat{U}^\top P_{2,1})} \\
&\leq \Pr[x_2 = x] \cdot \frac{1+\sqrt{5}}{2} \cdot \frac{\epsilon_{2,1}}{\min\left(\sigma_k(\widehat{P}_{2,1}), \sigma_k(\widehat{U}^\top P_{2,1})\right)^2} + \frac{\epsilon_{3,x,1}}{\sigma_k(\widehat{U}^\top P_{2,1})}
\end{aligned}$$

where the second inequality is from Lemma 12 and the last one uses the fact that

$$\|P_{3,x,1}\|_2 \leq \|P_{3,x,1}\|_F = \sqrt{\sum_{i,j}[P_{3,x,1}]_{i,j}^2} \leq \sum_{i,j}[P_{3,x,1}]_{i,j} = \Pr[x_2 = x].$$

Applying Lemma 21 as in the δ_∞ bound above, gives us the required result on Δ_x. Summing both sides over x results in the required bound on Δ.

δ_1 **bound:** For δ_1, we invoke Condition 4 to use the fact that $\|R\|_1 \leq 1$. Specifically,

$$\begin{aligned}
\delta_1 &= \left\|R(\widehat{U}^\top OR)^{-1}\widehat{U}^\top(\widehat{P}_1 - \vec{P}_1)\right\|_1 \\
&\leq \|R\|_1 \left\|(\widehat{U}^\top OR)^{-1}\widehat{U}^\top(\widehat{P}_1 - \vec{P}_1)\right\|_1 \quad \text{(norm consistency)} \\
&\leq \sqrt{k}\,\|R\|_1 \left\|(\widehat{U}^\top OR)^{-1}\widehat{U}^\top(\widehat{P}_1 - \vec{P}_1)\right\|_2 \quad (\|x\|_1 \leq \sqrt{n}\,\|x\|_2 \text{ for any } x \in \mathbb{R}^n) \\
&\leq \sqrt{k}\,\|R\|_1 \left\|(\widehat{U}^\top OR)^{-1}\widehat{U}^\top\right\|_2 \left\|(\widehat{P}_1 - \vec{P}_1)\right\|_2 \quad \text{(norm consistency)} \\
&\leq \sqrt{k}\,\|R\|_1 \left\|(\widehat{U}^\top OR)^{-1}\right\|_2 \cdot \epsilon_1 \quad (\text{defn. of } \epsilon_1,\ \widehat{U}^\top \text{ has orthogonal columns}) \\
&= \frac{\sqrt{k}\epsilon_1}{\sigma_k(\widehat{U}^\top OR)} \quad (\|R\|_1 \leq 1,\ \text{defn. of } L_2\text{-norm})
\end{aligned}$$

The desired bound on δ_1 is obtained by using Lemma 21. With $\varepsilon, \varepsilon_0$ as described in the above proof for δ_∞, we have that $\sigma_k(\widehat{U}^\top OR) \geq (\sqrt{3}/2)\sigma_k(U^\top OR)$. The required bound follows by plugging this inequality into the above upper bound for δ_1. □

A.1.4 Proof of Theorem 2

The following Lemmas 23 and 24 together with Lemmas 18,21,22 above, constitute the proof of Theorem 2 on joint probability accuracy. We state the results based on appropriate modifications of HKZ, and provide complete proofs. We also describe how the proofs generalize to the case of handling continuous observations using Kernel Density Estimation (KDE). First, define the following as in HKZ

$$\epsilon(i) = \min \left\{ \sum_{j \in S} \Pr[x_2 = j] : S \subseteq \{1 \ldots n\}, |S| = n - i \right\}$$

and let

$$n_0(\varepsilon) = \min\{i : \epsilon(i) \leq \varepsilon\}$$

The term $n_0(\varepsilon)$, which occurs in the theorem statement, can be interpreted as the minimum number of discrete observations that accounts for $1 - \epsilon$ of total marginal observation probability mass. Since this can be much lower than (and independent of) n in many applications, the analysis of HKZ is able to use n_0 instead of n in the sample complexity bound. This is useful in domains with large n, and our relaxation of HKZ preserves this advantageous property.

The following lemma quantifies how estimation errors accumulate while computing the joint probability of a length t sequence, due to errors in $\widehat{B}_x$ and $\widehat{b}$.

Lemma 23 *[Modification of HKZ Lemma 11] Assume $\widehat{U}^\top OR$ is invertible. For any time t:*

$$\sum_{x_{1:t}} \left\| R(U^\top OR)^{-1} \left(\widehat{B}_{x_{t:1}} \widehat{b}_1 - \widetilde{B}_{x_{t:1}} \widetilde{b}_1 \right) \right\|_1 \leq (1 + \Delta)^t \delta_1 + (1 + \Delta)^t - 1$$

PROOF: Proof by induction. The base case for $t = 0$, i.e. that $\left\| R(U^\top OR)^{-1} (\widehat{b}_1 - \widetilde{b}_1) \right\|_1 \leq \delta_1$ is true by definition of δ_1. For the rest, define unnormalized states $\widehat{b}_t = \widehat{b}_t(x_{1:t-1}) = \widehat{B}_{x_{t-1:1}} \widehat{b}_1$ and $\widetilde{b}_t = \widetilde{b}_t(x_{1:t-1}) = \widetilde{B}_{x_{t-1}:1} \widetilde{b}_1$. For some particular $t > 1$, assume the inductive hypothesis as follows

$$\sum_{x_{1:t}} \left\| R(\widehat{U}^\top OR)^{-1}\left(\widehat{b}_t - \widetilde{b}_t\right)\right\|_1 \leq (1+\Delta)^t\delta_1 + (1+\Delta)^t - 1$$

The sum over $x_{1:t}$ in the LHS can be decomposed as:

$$\begin{aligned}&\sum_{x_{1:t}} \left\| R(\widehat{U}^\top OR)^{-1}\left(\widehat{b}_t - \widetilde{b}_t\right)\right\|_1 \\ = \quad &\sum_{x}\sum_{x_{1:t-1}} \left\| R(U^\top OR)^{-1}\left((\widehat{B}_{x_t} - \widetilde{B}_{x_t})\widetilde{b}_t + (\widehat{B}_{x_t} - \widetilde{B}_{x_t})(\widehat{b}_t - \widetilde{b}_t) + \widetilde{B}_{x_t}(\widehat{b}_t - \widetilde{b}_t)\right)\right\|_1\end{aligned}$$

Using triangle inequality, the above sum is bounded by

$$\begin{aligned}&\sum_{x_t}\sum_{x_{1:t-1}} \left\| R(\widehat{U}^\top OR)^{-1}\left(\widehat{B}_{x_t} - \widetilde{B}_{x_t}\right)(\widehat{U}^\top O)\right\|_1 \left\| R(\widehat{U}^\top OR)^{-1}\widetilde{b}_t\right\|_1 \\ +&\sum_{x_t}\sum_{x_{1:t-1}} \left\| R(\widehat{U}^\top OR)^{-1}\left(\widehat{B}_{x_t} - \widetilde{B}_{x_t}\right)(\widehat{U}^\top O)\right\|_1 \left\| R(\widehat{U}^\top OR)^{-1}\left(\widehat{b}_t - \widetilde{b}_t\right)\right\|_1 \\ +&\sum_{x_t}\sum_{x_{1:t-1}} \left\| R(\widehat{U}^\top OR)^{-1}\widetilde{B}_t(\widehat{U}^\top OR)(\widehat{U}^\top OR)^{-1}\left(\widehat{b}_t - \widetilde{b}_t\right)\right\|_1\end{aligned}$$

Each of the above double sums is bounded separately. For the first, we note that $\left\| R(\widehat{U}^\top OR)^{-1}\widetilde{b}_t\right\|_1 = \Pr[x_{1:t-1}]$, which sums to 1 over $x_{1:t-1}$. The remainder of the double sum is bounded by Δ, by definition. For the second double sum, the inner sum over $\left\| R(\widehat{U}^\top OR)^{-1}(\widehat{b}_t - \widetilde{b}_t)\right\|_1$ is bounded using the inductive hypothesis. The outer sum scales this bound by Δ, by definition. Hence the second double sum is bounded by $\Delta((1+\Delta)^{t-1}\delta_1 + (1+\Delta)^{t-1} - 1)$. Finally, we deal with the third double sum as follows. We first replace $(\widehat{U}^\top OR)^{-1}\widetilde{B}_t(\widehat{U}^\top OR)$ by W_{x_t}, and note that $R \cdot W_{x_t} = A_{x_t}R$. Since A_{x_t} is entry-wise nonnegative by definition, $\|A_{x_t}\vec{v}\|_1 \leq \vec{1}_m^\top A_{x_t}|\vec{v}|$, where $|\vec{v}|$ denotes element-wise absolute value. Also note that $\vec{1}_m^\top \sum_{x_t} A_{x_t}|\vec{v}| = \vec{1}_m^\top T|\vec{v}| = \vec{1}_m^\top|\vec{v}| = \|\vec{v}\|_1$. Using this result with $\vec{v} = R(\widehat{U}^\top OR)^{-1}(\widehat{b}_t - \widetilde{b}_t)$ in the third double sum above, the inductive hypothesis bounds the double sum by

$(1+\Delta)^{t-1}\delta_1 + (1+\Delta)^{t-1} - 1$. Combining these three bounds gives us the required result:

$$\begin{aligned}
&\sum_{x_{1:t}} \left\| R(\widehat{U}^\top OR)^{-1} \left(\widehat{b}_t - \widetilde{b}_t\right) \right\|_1 \\
&\leq \Delta + \Delta((1+\Delta)^{t-1}\delta_1 + (1+\Delta)^{t-1} - 1) + (1+\Delta)^{t-1}\delta_1 + (1+\Delta)^{t-1} - 1 \\
&= \Delta + (1+\Delta)((1+\Delta)^{t-1}\delta_1 + (1+\Delta)^{t-1} - 1) \\
&= \Delta + (1+\Delta)^t\delta_1 + (1+\Delta)^t - 1 - \Delta \\
&= (1+\Delta)^t\delta_1 + (1+\Delta)^t - 1
\end{aligned}$$

thus completing the induction. □

The following lemma bounds the effect of errors in the normalizer $\widehat{b}_\infty$.

Lemma 24 *[Modification of HKZ Lemma 12] Assume $\epsilon_{2,1} \leq \sigma_k(P_{2,1})/3$. Then for any t,*

$$\sum_{x_{1:t}} \left| \Pr[x_{1:t}] - \widehat{\Pr}[x_{1:t}] \right| \leq (1+\delta_\infty)(1+\delta_1)(1+\Delta)^t - 1$$

PROOF: First note that the upper bound on $\epsilon_{2,1}$ along with Lemma 21, ensure that $\sigma_k(\widehat{U}^\top OR) > 0$ and so $(\widehat{U}^\top OR)$ is invertible. The LHS above can be decomposed into three sums that are dealt with separately:

$$\begin{aligned}
\sum_{x_{1:t}} \left| \Pr[x_{1:t}] - \widehat{\Pr}[x_{1:t}] \right| &= \sum_{x_{1:t}} \left| \widehat{b}_\infty^\top \widehat{B}_{x_{t:1}} \widehat{b}_1 - \vec{b}_\infty^\top B_{x_{t:1}} \vec{b}_1 \right| \\
&= \sum_{x_{1:t}} \left| \widehat{b}_\infty^\top \widehat{B}_{x_{t:1}} \widehat{b}_1 - \widetilde{b}_\infty^\top \widetilde{B}_{x_{t:1}} \widetilde{b}_1 \right| \\
&\leq \sum_{x_{1:t}} \left| (\widehat{b}_\infty - \widetilde{b}_\infty)^\top (\widehat{U}^\top OR)(\widehat{U}^\top OR)^{-1} \widetilde{B}_{x_{t:1}} \widetilde{b}_1 \right| \\
&\quad + \sum_{x_{1:t}} \left| (\widehat{b}_\infty - \widetilde{b}_\infty)^\top (\widehat{U}^\top OR)(\widehat{U}^\top OR)^{-1} (\widehat{B}_{x_{t:1}} \widehat{b}_1 - \widetilde{B}_{x_{t:1}} \widetilde{b}_1) \right| \\
&\quad + \sum_{x_{1:t}} \left| \widetilde{b}_\infty^\top (\widehat{U}^\top OR)(\widehat{U}^\top OR)^{-1} (\widehat{B}_{x_{t:1}} \widehat{b}_1 - \widetilde{B}_{x_{t:1}} \widetilde{b}_1) \right|
\end{aligned}$$

The first sum can be bounded as follows, using Hölders inequality and bounds from

Lemma 22:

$$\begin{aligned}\sum_{x_{1:t}}\left|(\widehat{b}_\infty-\widetilde{b}_\infty)^\top(\widehat{U}^\top OR)(\widehat{U}^\top OR)^{-1}\widetilde{B}_{x_{t:1}}\widetilde{b}_1\right| &\le \sum_{x_{1:t}}\left\|(\widehat{U}^\top O)^\top(\widehat{b}_\infty-\widetilde{b}_\infty)\right\|_\infty\left\|R(\widehat{U}^\top OR)^{-1}\widetilde{B}_{x_{t:1}}\widetilde{b}_1\right\|_1\\ &\le \sum_{x_{1:t}}\delta_\infty\left\|A_{x_{t:1}}\vec{\pi}\right\|_1\\ &= \sum_{x_{1:t}}\delta_\infty\Pr[x_{1:t}]\\ &= \delta_\infty\end{aligned}$$

The second sum can be bounded also using Hölders, as well as the bound in Lemma 23:

$$\begin{aligned}&\sum_{x_{1:t}}\left|(\widehat{b}_\infty-\widetilde{b}_\infty)^\top(\widehat{U}^\top OR)(\widehat{U}^\top OR)^{-1}(\widehat{B}_{x_{t:1}}\widehat{b}_1-\widetilde{B}_{x_{t:1}}\widetilde{b}_1)\right|\\ &\quad\le\left\|(\widehat{U}^\top O)^\top(\widehat{b}_\infty-\widetilde{b}_\infty)\right\|_\infty\left\|R(\widehat{U}^\top OR)^{-1}(\widehat{B}_{x_{t:1}}\widehat{b}_1-\widetilde{B}_{x_{t:1}}\widetilde{b}_1)\right\|_1\\ &\quad\le\delta_\infty((1+\Delta)^t\delta_1+(1+\Delta)^t-1)\end{aligned}$$

The third sum again uses Lemma 23:

$$\begin{aligned}\sum_{x_{1:t}}\left|\widetilde{b}_\infty^\top(\widehat{U}^\top OR)(\widehat{U}^\top OR)^{-1}(\widehat{B}_{x_{t:1}}\widehat{b}_1-\widetilde{B}_{x_{t:1}}\widetilde{b}_1)\right| &= \sum_{x_{1:t}}\left|\vec{1}^\top R(\widehat{U}^\top OR)^{-1}(\widehat{B}_{x_{t:1}}\widehat{b}_1-\widetilde{B}_{x_{t:1}}\widetilde{b}_1)\right|\\ &\le\left\|R(\widehat{U}^\top OR)^{-1}(\widehat{B}_{x_{t:1}}\widehat{b}_1-\widetilde{B}_{x_{t:1}}\widetilde{b}_1)\right\|_1\\ &\le(1+\Delta)^t\delta_1+(1+\Delta)^t-1\end{aligned}$$

Adding these three sums gives us:

$$\begin{aligned}\sum_{x_{1:t}}\left|\Pr[x_{1:t}]-\widehat{\Pr}[x_{1:t}]\right| &\le\delta_\infty+\delta_\infty((1+\Delta)^t\delta_1+(1+\Delta)^t-1)+(1+\Delta)^t\delta_1+(1+\Delta)^t-1\\ &\le\delta_\infty+(1+\delta_\infty)((1+\Delta)^t\delta_1+(1+\Delta)^t-1)\end{aligned}$$

which is the required bound. □

PROOF:(Theorem 2). Assume N and ε as in the theorem statement:

$$\varepsilon = \sigma_k(OR)\sigma_k(P_{2,1})\epsilon/(4t\sqrt{k})$$
$$N \geq C \cdot \frac{t^2}{\epsilon^2} \cdot \left(\frac{k}{\sigma_k(OR)^2\sigma_k(P_{2,1})^4} + \frac{k \cdot n_0(\varepsilon)}{\sigma_k(OR)^2\sigma_k(P_{2,1})^2} \right) \cdot \log(1/\eta)$$

First note that

$$\sum_{x_{1:t}} \left| \Pr[x_{1:t}] - \widehat{\Pr}[x_{1:t}] \right| \leq 2$$

since it is the L_1 difference between two stochastic vectors. Therefore, the theorem is vacuous for $\epsilon \geq 2$. Hence we can assume

$$\epsilon < 1$$

in the proof and let the constant C absorb the factor 4 difference due to the $1/\epsilon^2$ term in the expression for N.

The proof has three steps. We first list these steps then prove them below.

First step: for a suitable constant C, the following sampling error bounds follow from Lemma 18:

$$\epsilon_1 \leq \min\left(.05 \cdot (3/8) \cdot \sigma_k(P_{2,1}) \cdot \epsilon, .05 \cdot (\sqrt{3}/2) \cdot \sigma_k(OR) \cdot (1/\sqrt{k}) \cdot \epsilon\right) \quad \text{(A.14a)}$$
$$\epsilon_{2,1} \leq \min\left(.05 \cdot (1/8) \cdot \sigma_k(P_{2,1})^2 \cdot (\epsilon/5), .01 \cdot (\sqrt{3}/8) \cdot \sigma_k(OR) \cdot \sigma_k(P_{2,1})^2 \cdot (1/(t\sqrt{k})) \cdot \epsilon\right) \quad \text{(A.14b)}$$
$$\sum_x \epsilon_{3,x,1} \leq 0.39 \cdot (3\sqrt{3}/8) \cdot \sigma_k(OR) \cdot \sigma_k(P_{2,1}) \cdot (1/(t\sqrt{k})) \cdot \epsilon \quad \text{(A.14c)}$$

Second step: Lemma 22 together with equations (A.14) imply:

$$\delta_\infty \leq .05\epsilon \quad \text{(A.15a)}$$
$$\delta_1 \leq .05\epsilon \quad \text{(A.15b)}$$
$$\Delta \leq 0.4\epsilon/t \quad \text{(A.15c)}$$

Third step: By Lemma 24, equations (A.15) and the inequality

$$(1 + (a/t))^t \leq 1 + 2a \quad \text{for } a \leq 1/2 \quad \text{(A.16)}$$

we get the theorem statement.

Proof of first step: Note that for any value of matrix $P_{2,1}$, we can upper-bound $\sigma_k(P_{2,1})$ by 1:

$$\begin{aligned}
\sigma_k(P_{2,1}) &\leq \sigma_1(P_{2,1}) \\
&= \max_{\|x\|_2=1} \|P_{2,1}x\|_2 \\
&= \max_{\|x\|_2=1} \left(\sum_{j=1}^{n} \left(\sum_{i=1}^{n} [P_{2,1}]_{ij} x_i \right)^2 \right)^{1/2} \\
&\leq \max_{\|x\|_2=1} \sum_{j=1}^{n} \left| \sum_{i=1}^{n} [P_{2,1}]_{ij} x_i \right| \quad \text{(by norm inequality)} \\
&\leq \sum_{j=1}^{n} \sum_{i=1}^{n} |[P_{2,1}]_{ij}| \quad (|x_i| \leq 1 \text{ since } \|x\|_2 = 1) \\
&= \sum_{j=1}^{n} \sum_{i=1}^{n} [P_{2,1}]_{ij} \quad \text{(by non-negativity of } P_{2,1}) \\
&= 1 \quad \text{(by definition)}
\end{aligned}$$

Similarly, for any column-stochastic observation probability matrix O we can bound $\sigma_k(OR)$

by $\sqrt{k}$. First see that $\sigma_1(O) \leq \sqrt{m}$:

$$\begin{aligned}
\sigma_1(O) &= \max_{\|x\|_2=1} \|Ox\|_2 \\
&= \max_{\|x\|_2=1} \left(\sum_{j=1}^{m} \sum_{i=1}^{n} (O_{ij} x_i)^2 \right)^{1/2} \\
&\leq \max_{\|x\|_2=1} \left(\sum_{j=1}^{m} \sum_{i=1}^{n} O_{ij}^2 \right)^{1/2} \qquad (\|x\|_2 = 1 \Rightarrow |x_i| \leq 1) \\
&\leq \left(\sum_{j=1}^{m} (\sum_{i=1}^{n} O_{ij})^2 \right)^{1/2} \qquad \text{(by triangle inequality)} \\
&\leq \left(\sum_{j=1}^{m} 1^2 \right)^{1/2} \qquad \text{(by definition of } O\text{)} \\
&= \sqrt{m}
\end{aligned}$$

Now the bound on $\sigma_k(OR)$ follows from Condition 5 i.e. $\sigma_k(OR) \leq \sqrt{k/m}$:

$$\begin{aligned}
\sigma_k(OR) &= \min_{\|x\|_2=1} \|ORx\|_2 \\
&\leq \|O\|_2 \cdot \min_{\|x\|_2=1} \|Rx\|_2 \qquad \text{(by norm consistency)} \\
&\leq \sqrt{m} \min_{\|x\|_2=1} \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{k} (R_{ij} x_j)^2} \qquad (\because \|A\|_2 = \sigma_1(A) \text{ for any matrix } A)
\end{aligned}$$

Assume the c^{th} column of R obeys Condition 5 for some $1 \leq c \leq k$. Also assume $x = e_c$, the c^{th} column of the $k \times k$ identity matrix, which obeys the constraint $\|x\|_2 = 1$. Then every component of the inner sum is zero except when $j = c$, and the $\min$ expression can only get larger:

$$\begin{aligned}\sigma_k(OR) &\leq \sqrt{m}\sqrt{\sum_{i=1}^{m} R_{ic}^2} \\ &= \sqrt{m}\,\|R[\cdot,c]\|_2 \\ &\leq \sqrt{m}\sqrt{k/m} \\ &= \sqrt{k}\end{aligned}$$

hence proving that $\sigma_k(OR) \leq \sqrt{k}$.

Now we begin the proof with the ϵ_1 case. Choose a C that satisfies all previous bounds and also obeys $(\sqrt{C}/4) \cdot 0.05 \cdot (3/8) \geq 1$.

$$\epsilon_1 \leq \sqrt{1/N}(\sqrt{\ln(3/\eta)} + 1) \quad \text{(by Lemma 18)} \tag{A.17}$$

$$\leq \sqrt{1/N}(2\sqrt{\ln(3/\eta)}) \quad (\text{since } \sqrt{\ln(3/\eta)} \geq \sqrt{\ln 3} > 1) \tag{A.18}$$

Now, plugging in the assumed value of N:

$$\epsilon_1 \leq \frac{2\epsilon(\sigma_k(P_{2,1})^2\sigma_k(OR))}{t\sqrt{Ck(1+n_0(\varepsilon)\sigma_k(P_{2,1})^2)}}\sqrt{\frac{\ln(3/\eta)}{\ln(1/\eta)}} \tag{A.19}$$

Any substitutions that increase the right hand side of the above inequality preserve the inequality. We now drop the additive 1 in the denominator, replace $\sqrt{\ln(3/\eta)/\ln(1/\eta)}$ by 2 since it is at most $\sqrt{\ln 3}$, and drop the factors t, $\sqrt{n_0(\varepsilon)}$ from the denominator.

$$\epsilon_1 \leq \frac{4\sigma_k(OR)\sigma_k(P_{2,1})^2\epsilon}{\sqrt{Ck}\sigma_k(P_{2,1})} \tag{A.20}$$

$$\leq \frac{1}{\sqrt{C}} \cdot 4 \cdot \left[\sigma_k(OR)/\sqrt{k}\right] \cdot [\sigma_k(P_{2,1})] \cdot \epsilon$$

$$\leq \frac{1}{\sqrt{C}} \min\left(4 \cdot \sigma_k(P_{2,1}) \cdot \epsilon, 4 \cdot \sigma_k(OR) \cdot 1/\sqrt{k} \cdot \epsilon\right) \quad (\because \text{both } \left[\sigma_k(OR)/\sqrt{k}\right] \text{ and } [\sigma_k(P_{2,1})] \text{ are } \leq 1)$$

$$= \frac{1}{C'} \min\left(0.05 \cdot 3/8 \cdot \sigma_k(P_{2,1}) \cdot \epsilon, 0.05 \cdot \sqrt{3}/2 \cdot \sigma_k(OR) \cdot 1/\sqrt{k} \cdot \epsilon\right)$$

$$(\text{for } C' = \tfrac{\sqrt{C}}{4} \cdot 0.05 \cdot 3/8)$$

$$= \min\left(0.05 \cdot 3/8 \cdot \sigma_k(P_{2,1}) \cdot \epsilon, 0.05 \cdot \sqrt{3}/2 \cdot \sigma_k(OR) \cdot 1/\sqrt{k} \cdot \epsilon\right) \quad (\because C' \geq 1)$$

Hence proving the required bound for ϵ_1.

Next we prove the $\epsilon_{2,1}$ case. Choose a C that satisfies all previous bounds also obeys $(\sqrt{C}/4) \cdot 0.01 \cdot (\sqrt{3}/8) \geq 1$. Note that, since the bound on $\epsilon_{2,1}$ in Lemma 18 is the same as for ϵ_1, we can start with the analogue of equation (A.19):

$$\epsilon_{2,1} \leq \frac{2\epsilon(\sigma_k(P_{2,1})^2 \sigma_k(OR))}{t\sqrt{Ck(1+n_0(\varepsilon)\sigma_k(P_{2,1})^2)}} \sqrt{\frac{\ln(3/\eta)}{\ln(1/\eta)}}$$

We now drop the additive $n_0(\varepsilon)\sigma_k(P_{2,1})^2$ in the denominator, again replace $\sqrt{\ln(3/\eta)/\ln(1/\eta)}$ by 2 since it is at most $\sqrt{\ln 3}$, and drop the multiplicative factor t from the denominator.

$$\begin{aligned}
\epsilon_{2,1} &\leq \frac{1}{\sqrt{C}} \cdot 4 \cdot \sigma_k(P_{2,1})^2 \cdot \sigma_k(OR) \cdot (1/\sqrt{k}) \cdot \epsilon \\
&= \frac{1}{\sqrt{C}} \cdot 4 \cdot \left[\sigma_k(P_{2,1})^2\right] \cdot \left[\sigma_k(OR)/\sqrt{k}\right] \cdot \epsilon \\
&\leq \frac{1}{\sqrt{C}} \min\left(4 \cdot \sigma_k(P_{2,1})^2 \cdot \epsilon, 4 \cdot \sigma_k(OR) \cdot \sigma_k(P_{2,1})^2 \cdot 1/\sqrt{k} \cdot \epsilon\right) \quad (\because \left[\sigma_k(OR)/\sqrt{k}\right] \leq 1) \\
&\leq \frac{1}{C'} \min\left(0.05 \cdot 1/8 \cdot \sigma_k(P_{2,1})^2 \cdot \epsilon, 0.01 \cdot \sqrt{3}/8 \cdot \sigma_k(OR) \cdot \sigma_k(P_{2,1})^2 \cdot 1/\sqrt{k} \cdot \epsilon\right) \\
&\quad \text{(for } C' = (\sqrt{C}/4) \cdot 0.01 \cdot (\sqrt{3}/8)) \\
&\leq \min\left(0.05 \cdot 1/8 \cdot \sigma_k(P_{2,1})^2 \cdot \epsilon, 0.01 \cdot \sqrt{3}/8 \cdot \sigma_k(OR) \cdot \sigma_k(P_{2,1})^2 \cdot 1/\sqrt{k} \cdot \epsilon\right) \quad \text{(since } C' \geq 1)
\end{aligned}$$

hence proving the bound on $\epsilon_{2,1}$.

Finally for $\sum_x \epsilon_{3,x,1}$, assume C such that $\frac{2 \cdot 0.39 \cdot (3\sqrt{3}/8)\sqrt{C}}{16+\sqrt{C}} \geq 1$ in addition to previous requirements on C. we first restate the bound from Lemma 18:

$$\begin{aligned}
\sum_x \epsilon_{3,x,1} &\leq \min_j \left(\sqrt{j/N}\left(\sqrt{\ln 3/\eta}+1\right) + 2\epsilon(j)\right) + \sqrt{1/N}\left(\sqrt{\ln 3/\eta}+1\right) \\
&\leq \sqrt{n_0(\varepsilon)/N}\left(\sqrt{\ln 3/\eta}+1\right) + 2\epsilon(n_0(\varepsilon)) + \sqrt{1/N}\left(\sqrt{\ln 3/\eta}+1\right) \\
&\leq \sqrt{1/N}\left(\sqrt{\ln 3/\eta}+1\right)(n_0(\varepsilon)+1) + 2\varepsilon \quad \text{(since } \epsilon(n_0(\varepsilon)) \leq \varepsilon)
\end{aligned}$$

The first two terms are exactly as before, so we perform the same steps as in equations (A.17)-(A.20) except we do not drop $t\sqrt{n_0(\varepsilon)}$, to get:

$$\begin{aligned}
\sum_x \epsilon_{3,x,1} &\leq \frac{4\sigma_k(OR)\sigma_k(P_{2,1})^2\epsilon}{\sqrt{Ckn_0(\varepsilon)}\sigma_k(P_{2,1})}\left(n_0(\varepsilon)+1\right) + 2\varepsilon \\
&\leq \frac{4\sigma_k(OR)\sigma_k(P_{2,1})\epsilon}{t\sqrt{Ckn_0(\varepsilon)}} \cdot (2 \cdot n_0(\varepsilon)) + 2\sigma_k(OR)\sigma_k(P_{2,1})\epsilon/4t\sqrt{k} \\
&\quad \text{(since } 1 + n_0(\varepsilon) \leq 2 \cdot n_0(\varepsilon)\text{, and plugging in } \varepsilon\text{)} \\
&\leq \sigma_k(OR) \cdot \sigma_k(P_{2,1}) \cdot t\sqrt{k} \cdot \epsilon \cdot \left(8/\sqrt{C} + 1/2\right) \\
&\leq \frac{1}{C'} 0.39 \cdot (3\sqrt{3}/8) \cdot \sigma_k(OR) \cdot \sigma_k(P_{2,1}) \cdot t\sqrt{k} \cdot \epsilon \quad \text{(for } C' = \tfrac{2 \cdot 0.39 \cdot (3\sqrt{3}/8)\sqrt{C}}{16+\sqrt{C}}\text{)} \\
&\leq 0.39 \cdot (3\sqrt{3}/8) \cdot \sigma_k(OR) \cdot \sigma_k(P_{2,1}) \cdot t\sqrt{k} \cdot \epsilon \quad \text{(since } C' > 1 \text{ by assumption)}
\end{aligned}$$

Hence proving the required bound for $\sum_x \epsilon_{3,x,1}$.

Proof of second step: Substituting from equation (A.14) into δ_1 in Lemma 22:

$$\begin{aligned}
\delta_1 &\leq \frac{2}{\sqrt{3}} \frac{\sqrt{k}}{\sigma_k(OR)} \cdot \epsilon_1 \\
&\leq \frac{2}{\sqrt{3}} \frac{\sqrt{k}}{\sigma_k(OR)} \min\left(.05 \cdot \frac{3}{8}\sigma_k(P_{2,1})\epsilon, .05 \cdot \frac{\sqrt{3}}{2}\sigma_k(OR)\frac{1}{\sqrt{k}}\epsilon\right) \\
&= .05\epsilon \cdot \min\left(\frac{\sqrt{3}}{4}\frac{\sqrt{k}}{\sigma_k(OR)}\sigma_k(P_{2,1}), 1\right) \\
&\leq .05\epsilon
\end{aligned}$$

Substituting from equation (A.14) into δ_∞ in Lemma 22:

$$\begin{aligned}
\delta_\infty &\leq 4\left(\frac{\epsilon_{2,1}}{\sigma_k(P_{2,1})^2} + \frac{\epsilon_1}{3\sigma_k(P_{2,1})}\right) \\
&\leq \frac{4}{\sigma_k(P_{2,1})^2}\min\Big(.05\cdot(1/8)\cdot\sigma_k(P_{2,1})^2\cdot(\epsilon/5), .01\cdot(\sqrt{3}/8)\cdot\sigma_k(OR)\cdot\sigma_k(P_{2,1})^2\cdot(1/(t\sqrt{k}))\cdot\epsilon\Big) \\
&\quad + \frac{4}{3\sigma_k(P_{2,1})}\min\Big(.05\cdot(3/8)\cdot\sigma_k(P_{2,1})\cdot\epsilon, .05\cdot(\sqrt{3}/2)\cdot\sigma_k(OR)\cdot(1/\sqrt{k})\cdot\epsilon\Big) \\
&\leq \min\Big(.05\epsilon, .04\cdot(\sqrt{3}/8)\cdot\sigma_k(OR)\cdot(1/(t\sqrt{k}))\cdot\epsilon\Big) \\
&\quad + \min\left(.05\cdot(1/2)\cdot\epsilon, .05\cdot(2/\sqrt{3})\cdot\frac{\sigma_k(OR)}{\sigma_k(P_{2,1})}\cdot(1/\sqrt{k})\cdot\epsilon\right) \\
&\leq .05\epsilon(.01+.5) \\
&\leq .05\epsilon
\end{aligned}$$

Substituting from equation (A.14) into Δ in Lemma 22:

$$\begin{aligned}
\Delta &\leq \frac{8}{\sqrt{3}}\cdot\frac{\sqrt{k}}{\sigma_k(OR)}\cdot\left(\frac{\epsilon_{2,1}}{\sigma_k(P_{2,1})^2} + \frac{\Sigma_x\epsilon_{3,x,1}}{3\sigma_k(P_{2,1})}\right) \\
&\leq \frac{8\sqrt{k}}{\sqrt{3}\sigma_k(OR)}\cdot\left(\frac{1}{\sigma_k(P_{2,1})^2}\min\left(.05\cdot(1/8)\cdot\sigma_k(P_{2,1})^2\cdot(\epsilon/5), .01\cdot(\sqrt{3}/8)\cdot\sigma_k(OR)\cdot\sigma_k(P_{2,1})^2\cdot\frac{\epsilon}{t\sqrt{k}}\right)\right. \\
&\quad \left. + \frac{1}{3\sigma_k(P_{2,1})}0.39\cdot(3\sqrt{3}/8)\cdot\sigma_k(OR)\cdot\sigma_k(P_{2,1})\cdot(1/(t\sqrt{k}))\cdot\epsilon\right) \\
&= \left(\min\left(.05\cdot(\epsilon/5)\frac{\sqrt{k}}{\sqrt{3}\sigma_k(OR)}, .01\cdot\frac{\epsilon}{t}\right) + 0.39\cdot\frac{\epsilon}{t}\right) \\
&\leq .01\cdot\frac{\epsilon}{t} + 0.39\cdot\frac{\epsilon}{t} \\
&\leq 0.4\epsilon/t
\end{aligned}$$

Proof of third step: By Lemma 24,

$$\begin{aligned}
\sum_{x_{1:t}} \left|\Pr[x_{1:t}] - \widehat{\Pr}[x_{1:t}]\right| &\leq (1+\delta_\infty)(1+\delta_1)(1+\Delta)^t - 1 \\
&\leq (1+.05\epsilon)(1+.05\epsilon)(1+0.4\epsilon/t)^t - 1 \quad \text{(by equations (A.15))} \\
&\leq (1+.05\epsilon)(1+.05\epsilon)(1+0.8\epsilon) - 1 \quad \text{(by equation (A.16), since } 0.4\epsilon < 1/2\text{)} \\
&= 1 + .05\epsilon + .05\epsilon + .05^2\epsilon + 0.8\epsilon + .04\epsilon^2 + .04\epsilon^2 + (.05)^2 \cdot .08\epsilon^3 - 1 \\
&= .0002\epsilon^3 + .0825\epsilon^2 + 0.9\epsilon \\
&\leq (.0002 + .0825 + 0.9)\epsilon \quad \text{(since } \epsilon < 1 \text{ by assumption)} \\
&= 0.9827\epsilon \\
&< \epsilon
\end{aligned}$$

This completes the proof of Theorem 2. □

A.1.5 Proof of Theorem 2 for Continuous Observations

For continuous observations, we use Kernel Density Estimation (KDE) [109] to model the observation probability density function (PDF). We use a fraction of the training data points as kernel centers, placing one multivariate Gaussian kernel at each point.[1] The KDE estimator of the observation PDF is a convex combination of these kernels; since each kernel integrates to 1, this estimator also integrates to 1. KDE theory [109] tells us that as the number of kernel centers and the number of samples go to infinity and the kernel bandwidth goes to zero (at appropriate rates), the KDE estimator converges to the observation PDF in L_1 norm. The kernel density estimator is completely determined by the normalized vector of kernel weights; therefore, if we can estimate this vector accurately, our estimate will converge to the observation PDF as well.

Hence our goal is to predict the correct expected value of this normalized kernel vector given all past observations (or more precisely, given the appropriate sequence of past

[1]We use a general elliptical covariance matrix, chosen by SVD: that is, we use a spherical covariance after projecting onto the singular vectors and scaling by the square roots of the singular values.

observations, or the appropriate indicative events/features). In the context of Theorem 2, joint probability estimates for t-length observation sequences are effectively the expectation of entries in a t-dimensional tensor formed by the outer product of t indicator vectors. When we move to KDE, we instead estimate the expected outer product of t *stochastic* vectors, namely, the normalized kernel weights at each time step. As long as the sum of errors in estimating entries of this table goes to zero for any fixed t as the number of samples increases, our estimated observation PDFs will have bounded error.

The only differences in the proof are as follows. In Lemma 18, we observe $\vec{q_i}$ to be stochastic vectors instead of indicator vectors; their expectation is still the true value of the quantity we are trying to predict. $\vec{p_i}$ are also stochastic vectors in that proof. In the proof of Proposition 20, p_k is an arbitrary stochastic vector. Also, $\vec{q_i}^\top \vec{q_i} \leq \|\vec{q_i}\|_1 = 1$ now instead of being always equal to 1, and the same holds for $\vec{p_i}^\top \vec{p_i}$. Also $\|\widehat{p_i} - \vec{p_i}\|_2 \leq \|\widehat{p_i} - \vec{p_i}\|_1 = 1$ (by triangle inequality). Besides these things, the above proof goes through as it is.

Note that in the continuous observation case, there are continuously many observable operators W_x that can be computed. We compute one base operator for each kernel center, and use convex combinations of these base operators to compute observable operators as needed.

A.2 Learning with Ambiguous Observations: Example

When stacking observations, the modified, larger $\overline{P_{2,1}} \in \mathbb{R}^{\overline{n} \times \overline{n}}$ still has rank at most k since it can be written in the form $\overline{P_{2,1}} = GTH$ for some matrices $G, H^\top \in \mathbb{R}^{\overline{n} \times m}$. For example, if $n = 2$ for an HMM with ambiguous observations, and we believe stacking 2 observations per timestep will yield a sufficiently informative observation, the new observation space will consist of all $\overline{n} = n^2 = 4$ possible tuples of single observations and $P_{2,1} \in \mathbb{R}^{n^2 \times n^2}$, with each observation i corresponding to a tuple $< i_1, i_2 >$ of the original observations. Specifically,

$$\begin{aligned}
\overline{P_{2,1}}(j,i) &= \Pr(x_4 = j_2, x_3 = j_1, x_2 = i_2, x_1 = i_1) \\
&= \sum_{a,b,c,d} \Pr(x_4 = j_2, x_3 = j_1, x_2 = i_2, x_1 = i_1, h_4 = d, h_3 = c, h_2 = b, h_1 = a) \\
&= \sum_{a,b,c,d} O_{j_2 d} T_{dc} O_{j_1 c} T_{cb} O_{i_2 b} T_{ba} O_{i_1 a} \pi_a \\
&= \sum_{b,c} \overline{O}_{j,c} T_{cb} [\ \mathrm{diag}(\pi) \overline{O}^{\mathsf{T}}]_{b,i} \quad \text{where } \overline{O}_{j,c} = \sum_{d} O_{j_2 d} T_{dc} O_{j_1 c} \\
\Rightarrow \overline{P_{2,1}} &= \overline{O} T\ \mathrm{diag}(\pi) \overline{O}^{\mathsf{T}}
\end{aligned}$$

Similarly, we can show that $\overline{P_{3,x,1}} = GTH^{\mathsf{T}}$ for some matrices $G, H^{\mathsf{T}} \in \mathbb{R}^{\overline{n} \times m}$. The exact formulae will differ for different choices of past and future observable statistics.

A.3 Synthetic Example RR-HMM Parameters

Example 1

$$T = \begin{bmatrix} 0.3894 & 0.2371 & 0.3735 \\ 0.2371 & 0.4985 & 0.2644 \\ 0.3735 & 0.2644 & 0.3621 \end{bmatrix} \quad O = \begin{bmatrix} 0.6000 & 0.2000 & 0.2000 \\ 0.2000 & 0.6000 & 0.2000 \\ 0.2000 & 0.2000 & 0.6000 \end{bmatrix} \tag{A.21}$$

Example 2

$$T = \begin{bmatrix} 0.6736 & 0.0051 & 0.1639 \\ 0.0330 & 0.8203 & 0.2577 \\ 0.2935 & 0.1746 & 0.5784 \end{bmatrix} \quad O = \begin{bmatrix} 1 & 0 & .5 \\ 0 & 1 & .5 \end{bmatrix}$$

Example 3

$$T = \begin{bmatrix} 0.7829 & 0.1036 & 0.0399 & 0.0736 \\ 0.1036 & 0.4237 & 0.4262 & 0.0465 \\ 0.0399 & 0.4262 & 0.4380 & 0.0959 \\ 0.0736 & 0.0465 & 0.0959 & 0.7840 \end{bmatrix} \quad O = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

A.4 Consistency Result for Learning with Indicative and Characteristic Features

Here we prove *consistency* of the RR-HMM learning algorithm when handling arbitrary continuous features of groups of observations, termed *indicative features* for the past and *characteristic features* for the future. As before, the low-rank transition matrix is denoted by $T = RS$. The observation probability model is O, which we think of as a ”matrix” whose first dimension is continuous (corresponding to observations x_t) and whose second dimension is number of states, so that ”columns” are probability distributions. The statistics are W^P, W, and W^F (for past, present, and future), each of which we think of a ”matrix” whose first dimension is (number of statistics) and whose second is continuous (corresponding to observations), so that ”rows” are statistics that we collect. W_j means the j^{th} row of W. The initial distribution over states, as before, is π (a vector of length number of states). In this scenario, the quantities $\vec{P}_1, P_{2,1}$ and $P_{3,x,1}$ no longer contain probabilities but rather *expected values* of singletons, pairs and triples of *features*. For the special case of features that are indicator variables of *events*, we recover the conventional case where $\vec{P}_1, P_{2,1}, P_{3,x,1}$ consist of probabilities, and our performance bounds hold in addition to the consistency results which we show below for the general features case. In the derivation below, we consider $P_{3,x,1}$ to be a three-dimensional *tensor* rather than a series of matrices, for simplicity.

With this notation, we have the following expressions for $P_{2,1}$ and $P_{3,x,1}$ (see Derivations subsection below):

$$P_{2,1} = (W^P OR)(S \ \mathrm{diag}(\pi) O' {W^F}^{\mathsf{T}}) \tag{A.22}$$

$$P_{3,x,1}(:, j, :) = (W^P OR)(S \ \mathrm{diag}(W_j O) R)(S \ \mathrm{diag}(\pi) O' {W^F}^{\mathsf{T}}) \tag{A.23}$$

$$\tag{A.24}$$

Factoring $P_{2,1} = UV$, we know that U has the same range (column span) as $P_{2,1}$, so

$$U = (W^P OR)A$$

for some (square, invertible) matrix A in the low-rank space. Assuming $W^P OR$ has full column rank, and writing U^+ for the pseudo-inverse of U, we get

$$U^+ P_{2,1} = A^{-1} S \ \mathrm{diag}(\pi) O' {W^F}^\mathsf{T} \tag{A.25}$$

$$U^+ P_{3,x,1}(:,j,:) = A^{-1}(S \ \mathrm{diag}(W_j O) R)(S \ \mathrm{diag}(\pi) O' {W^F}^\mathsf{T}) \tag{A.26}$$

Assuming $S \ \mathrm{diag}(\pi) O' {W^F}^\mathsf{T}$ has full row rank, we then get

$$B_j \equiv U^+ P_{3,x,1}(:,j,:)(U^+ P_{2,1})^+ = A^{-1}(S \ \mathrm{diag}(W_j O) R) A$$

This is the analog of the original result about our learned observable operators: B_j is a similarity transform away from $S \ \mathrm{diag}(W_j O) R$. However, these new ”observable operators” aren’t necessarily what we need for tracking, since $W_j O$ isn’t necessarily a probability distribution. It’s possible that we can either come up with conditions under which the new ”observable operators” are really what we need for tracking, or some slight modification of the statistics collected that makes them so.

Derivations

Derivations for $P_{2,1}, P_{3,x,1}$ that also show them to be low-rank:

$$\begin{aligned}
P_{2,1}(i,k) &= \mathbb{E}(t_i^P(x_t)t_k^F(x_{t+1})) \\
&= \int\int \mathbb{P}(x_t, x_{t+1})t_i^P(x_t)t_k^F(x_{t+1})dx_t dx_{t+1} \\
&= \int\int \sum_{h_t}\sum_{h_{t+1}} \mathbb{P}(h_t)T(h_t, h_{t+1}) \\
&\quad O(x_t, h_t)O(x_{t+1}, h_{t+1})t_i^P(x_t)t_k^F(x_{t+1})dx_t dx_{t+1} \\
&= \int\int \sum_{h_t}\sum_{h_{t+1}}\sum_{z_t} \mathbb{P}(h_t)R(h_t, z_t)S(z_t, h_{t+1}) \\
&\quad O(x_t, h_t)O(x_{t+1}, h_{t+1})t_i^P(x_t)t_k^F(x_{t+1})dx_t dx_{t+1} \\
&= \sum_{z_t}\left(\int\sum_{h_t} \mathbb{P}(h_t)R(h_t, z_t)O(x_t, h_t)t_i^P(x_t)dx_t\right) \\
&\quad \left(\int\sum_{h_{t+1}} S(z_t, h_{t+1})O(x_{t+1}, h_{t+1})t_k^F(x_{t+1})dx_{t+1}\right)
\end{aligned}$$

For $P_{3,x,1}$:

$$\begin{aligned}
P_{3,x,1}(i,j,k) &= \mathbb{E}(t_i^P(x_{t-1})t_j(x_t)t_k^F(x_{t+1})) \\
&= \int\int\int \mathbb{P}(x_{t-1},x_t,x_{t+1})t_i^P(x_{t-1})t_j(x_t)t_k^F(x_{t+1})dx_{t-1}dx_tdx_{t+1} \\
&= \int\int\int \sum_{h_{t-1}}\sum_{h_t}\sum_{h_{t+1}} \mathbb{P}(h_{t-1})T(h_{t-1},h_t)T(h_t,h_{t+1}) \\
&\quad O(x_{t-1},h_{t-1})O(x_t,h_t)O(x_{t+1},h_{t+1})t_i^P(x_{t-1})t_j(x_t)t_k^F(x_{t+1})dx_{t-1}dx_tdx_{t+1} \\
&= \int\int\int \sum_{h_{t-1}}\sum_{h_t}\sum_{h_{t+1}}\sum_{z_{t-1}}\sum_{z_t} \mathbb{P}(h_{t-1})R(h_{t-1},z_{t-1})S(z_{t-1},h_t)R(h_t,z_t)S(z_t,h_{t+1}) \\
&\quad O(x_{t-1},h_{t-1})O(x_t,h_t)O(x_{t+1},h_{t+1})t_i^P(x_{t-1})t_j(x_t)t_k^F(x_{t+1})dx_{t-1}dx_tdx_{t+1} \\
&= \sum_{z_{t-1}}\sum_{z_t}\left(\int\sum_{h_{t-1}}\mathbb{P}(h_{t-1})R(h_{t-1},z_{t-1})O(x_{t-1},h_{t-1})t_i^P(x_{t-1})dx_{t-1}\right) \\
&\quad \left(\int\sum_{h_t}S(z_{t-1},h_t)R(h_t,z_t)O(x_t,h_t)t_j(x_t)dx_t\right) \\
&\quad \left(\int\sum_{h_{t+1}}S(z_t,h_{t+1})O(x_{t+1},h_{t+1})t_k^F(x_{t+1})dx_{t+1}\right)
\end{aligned}$$

A.5 Consistency Result for Learning PSRs

In this case, R and S don't exist in the true model, and there is no matrix O. There are only *observable operators* [69] which we will call M_i, a normalization vector b_∞, and from these we can derive update vectors $b_i^\mathsf{T} = b_\infty^\mathsf{T} M_i$. Assume for now that the observation features are single discrete observations, and that there is only a single action. The tests (characteristic events) are assumed to be indicator functions of the next single observation so the expected value of the test is the probability of seeing that observation. The histories (indicative events) similarly corresponds to the single previous observation, i.e. the PSR/OOM is assumed to be 1-step observable (see [73] for a generalization of this proof to general indicative and characteristic events, multiple control inputs and continuous observations). Assume i, j, k are the indices of observations x_t, x_{t+1}, x_{t+2} respectively. Let z denote the hidden state at time t, $P(z)$ denote the belief over z, and $\bar{z}$ denote $\int zP(z)dz$. For P_3:

$$\begin{aligned}
P_3(i,j,k) &= \int P(z)P(x_t = i \mid h_t = z)P(x_{t+1} = j \mid h_t = z)P(x_{t+2} = k \mid x_{t+1} = j, h_t = z) \\
&= \int P(z)\left(b_i^\mathsf{T} z\right)\left(b_j^\mathsf{T} \frac{M_i z}{b_i^\mathsf{T} z}\right)\left(b_k^\mathsf{T} \frac{M_j \frac{M_i z}{b_i^\mathsf{T} z}}{b_j^\mathsf{T} \frac{M_i z}{b_i^\mathsf{T} z}}\right) dz \\
&= \int P(z) b_k^\mathsf{T} M_j M_i z dz \\
&= b_k^\mathsf{T} M_j M_i \bar{z}
\end{aligned}$$

Similarly, for P_2:

$$\begin{aligned}
P_2(i,j) &= \int P(z)P(x_t = i \mid h_t = z)P(x_{t+1} = j \mid h_t = z) \\
&= \int P(z) b_j^\mathsf{T} M_i z dz \\
&= b_j^\mathsf{T} M_i \bar{z}
\end{aligned}$$

Let B and M be matrices such that:

$$B = \begin{bmatrix} \vdots & \vdots & \vdots \\ -- & b_j^\mathsf{T} & -- \\ \vdots & \vdots & \vdots \end{bmatrix}_{n \times k} \quad \text{and} \quad M = \begin{bmatrix} \cdots & | & \cdots \\ \cdots & M_i\bar{z} & \cdots \\ \cdots & | & \cdots \end{bmatrix}_{k \times n}$$

Then, we can write

$$P_2 = BM$$
$$P_3(:, j, :) = BM_jM$$

Factoring $P_2 = UV$, we know that U has the same range (column span) as P_2, so

$$U = BA$$

for some (square, invertible) matrix A in the low-rank space. Assuming B has full column rank, and writing U^+ for the pseudo-inverse of U, we get

$$U^+ P_2 = A^{-1} M \tag{A.27}$$

$$U^+ P_3(:, j, :) = A^{-1} M_j M \tag{A.28}$$

Assuming M has full row rank, we then get

$$B_j \equiv U^+ P_3(:, j, :)(U^+ P_2)^+ = A^{-1} M_j A$$

B_j is a similarity transform away from M_j, the OOM observable operator.

Bibliography

[1] M. W. Kadous. *Temporal classification: Extending the classification paradigm to multivariate time series.* PhD thesis, University of New South Wales, 2002. (document), 4.4.5, 4.6

[2] Seth L. Lacy and Dennis S. Bernstein. Subspace identification with guaranteed stability using constrained optimization. In *Proc. American Control Conference*, 2002. (document), 5.1, 5.2, 5.1, 5.4

[3] B. Boots. Learning Stable Linear Dynamical Systems. Data Analysis Project, Carnegie Mellon University, 2009. (document), 3.3.2, 5.4.2, 5.4

[4] L. R. Rabiner. A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proc. IEEE*, 77(2):257–285, 1989. 2, 2.2, 4.4, 4.5, 6, 6.1

[5] L. R. Bahl, F. Jelinek, and R. L. Mercer. A maximum likelihood approach to continouous speech recognition. In *IEEE Trans Pattern Anal Machine Intell.*, volume PAMI-5, pages 179–190, 1983. 2

[6] P. Boufounos S. El-Difrawy and D. Ehrlich. Hidden Markov Models for DNA Sequencing. In *Proc. of Workshop on Genomic Signal Processing and Statistics (GENSIPS)*, October 2002. 2

[7] M. Brand, N. Oliver, and A. Pentland. Coupled hidden Markov models for complex action recognition. In *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 1997. 2, 2.5

[8] K. Seymore, A. McCallum, and R. Rosenfeld. Learning hidden Markov model structure for information extraction. In *AAAI'99 Wkshp Machine Learning for Information Extraction*, 1999. 2

[9] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, IT-13:260–267, 1967. 2.3

[10] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995. 2.4

[11] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons Inc, 1973. 2.4

[12] Daniel Hsu, Sham Kakade, and Tong Zhang. A spectral algorithm for learning hidden markov models. In *COLT*, 2009. 2.4, 6.1, 6.2, 6.2.1, 6.5

[13] Daniel Hsu, Sham Kakade, and Tong Zhang. A spectral algorithm for learning hidden markov models. *http://arxiv.org/abs/0811.4413*, 2008. 2.4

[14] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal Royal Statistical Society, Series B*, 39:1–38, 1977. 2.4.1

[15] L. Baum. An inequality and associated maximization technique in statistical estimation of probabilistic functions of a Markov process. *Inequalities*, 3:1–8, 1972. 2.4.1, 6.2

[16] Zoubin Ghahramani. An introduction to hidden markov models and bayesian networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 15:9–42, 2001. 2.5, 3.5

[17] Yoshua Bengio and Paolo Frasconi. An Input Output HMM Architecture. In *Advances in Neural Information Processing Systems*, 1995. 2.5, 6.5

[18] Shai Fine, Yoram Singer, and Naftali Tishby. The Hierarchical Hidden Markov Model: Analysis and Applications. *Machine Learning*, 32:41–62, 1998. 2.5

[19] Z. Ghahramani and M. I. Jordan. Factorial hidden Markov models. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Proc. Conf. Advances in Neural Information Processing Systems, NIPS*, volume 8, pages 472–478. MIT Press, 1995. 2.5

[20] S. Roweis. Constrained Hidden Markov Models. In *Advances in Neural Information Processing Systems (NIPS)*, volume 12. MIT Press, 2000. 2.5

[21] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006. 2.5, 7.1.1

[22] M. J. Beal, Z. Ghahramani, and C. E. Rasmussen. The Infinite Hidden Markov Model. In Becker S. Dietterich, T.G. and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 14, pages 577–585. MIT Press, 2002. 2.5, 7.1.1

[23] Emily B. Fox, Erik Sudderth, Michael I. Jordan, and Alan Willsky. An HDP-HMM for systems with state persistence. In *Proc. International Conference on Machine Learning*, July 2008. 2.5

[24] R.E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 1960. 3, 3.2

[25] P. Brockwell and R. A. Davis. *Time Series: Theory and Methods*. Springer, 1991. 3, 3.5

[26] L. Ljung. *System Identification: Theory for the user*. Prentice Hall, 2nd edition, 1999. 3, 3.2, 3.3.2, 3.3.2, 5.2

[27] P. Van Overschee and B. De Moor. *Subspace Identification for Linear Systems: Theory, Implementation, Applications*. Kluwer, 1996. 3, 3.2, 3.3.2, 3.3.2, 3.3.2, 3.3.2, 3.3.2, 5.2, 6.1, 6.2, 6.2.1, 6.2.4, 6.4.1

[28] Zoubin Ghahramani and Geoffrey E. Hinton. Parameter estimation for Linear Dynamical Systems. Technical Report CRG-TR-96-2, U. of Toronto, Department of Comp. Sci., 1996. 3, 6

[29] Tohru Katayama. *Subspace Methods for System Identification: A Realization Approach*. Springer, 2005. 3, 3.2, 3.3.2, 6.2, 6.2.4, 6.4.1

[30] H. Rauch. Solutions to the linear smoothing problem. In *IEEE Transactions on Automatic Control*, 1963. 3.2

[31] Kevin Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, UC Berkeley, 2002. 3.2

[32] Roger Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985. 3.3.1, 3.3.2, 3.3.2, 2, 5.3.2, 6.2

[33] Aristide Halanay and Vladimir Răsvan. *Stability and Stable Oscillations in Discrete Time Systems*. CRC, 2000. 3.4

[34] R. H. Shumway and D. S. Stoffer. An approach to time series smoothing and forecasting using the EM algorithm. *Journal of Time Series Analysis*, 3(4), 1982. 3.5

[35] R. E. Kopp and R. J. Orford. Linear regression applied to system identification and adaptive control systems. *Journal of the American Institute of Aeronautics and Astronautics*, 1:2300–2306, 1963. 3.5

[36] H. Cox. On the estimation of state variables and parameters for noisy dynamic systems. *IEEE Transactions on Automatic Control*, 9:5–12, 1964. 3.5

[37] Cen Li and Gautam Biswas. Temporal pattern generation using hidden markov model based unsupervised classification. volume 1642, pages 245–256, 1999. 4.1, 4.2, 4.3, 4.3.3

[38] M. Ostendorf and H. Singer. Hmm topology design using maximum likelihood successive state splitting. *Computer Speech and Language*, 11:17–41, 1997. 4.1, 4.2, 4.3

[39] Matthew Brand. Structure learning in conditional probability models via an entropic prior and parameter extinction. *Neural Computation*, 11(5):1155–1182, 1999. 4.1, 4.2

[40] Sajid Siddiqi, Geoffrey J. Gordon, and Andrew Moore. Fast state discovery for HMM model selection and learning. In *Proc. AISTATS*, 2007. 4.1

[41] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 1978. 4.1, 4.3.3

[42] A. Stolcke and S. Omohundro. Best-first Model Merging for Hidden Markov Model Induction. Technical Report TR-94-003, Intl. Computer Science Institute, 1994. 4.2, 4.3, 4.3.3, 4.6

[43] Hermann Ney. Acoustic modeling of phoneme units for continuous speech recognition. In *Proc.* 5^{th} *Europ. Signal Processing Conference*, 1990. 4.3, 4.3.3

[44] R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, 1998. 4.3

[45] Dan Geiger, David Heckerman, and Christopher Meek. Asymptotic Model Selection for Directed Networks with Hidden Variables. Technical Report MSR-TR-96-07, Microsoft Research, 1999. 4.3.3, 4.6

[46] D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. UCI repository of machine learning databases, 1998. 4.4.1

[47] Andrew Howard and Nicholas Roy. The robotics data set repository (radish), 2003. 4.4.1

[48] Sajid M. Siddiqi and Andrew W. Moore. Fast inference and learning in large-state-space HMMs. In *Proc. ICML*, 2005. 4.4.1, 7.1.1, 7.1.1

[49] Liu Ren, Alton Patrick, Alexei A. Efros, Jessica K. Hodgins, and James M. Rehg. A data-driven approach to quantifying natural human motion. *SIGGRAPH 2005*, 24(3):1090–1097, August 2005. 4.4.1

[50] T. Starner and A. Pentland. Visual Recognition of American Sign Language Using Hidden Markov Models. In *Proc., Intl. Workshop on Automatic Face and Gesture Recognition (IWAFGR)*, 1995. 4.4.5

[51] Y. Liu, Q. Xiang, Y. Wang, and L. Cai. Cultural style based music classification of audio signals. In *ICASSP*, 2009. 4.5

[52] D. Rybach, C. Gollam, R. Schluter, and H. Ney. Audio segmentation for speech recognition using segment features. In *ICASSP*, 2009. 4.5

[53] J. Portelo, M. Bugalho, I. Trancoso, J. Neto, A. Abad, and A. Serralheiro. Non-speech audio event detection. In *ICASSP*, 2009. 4.5

[54] S. Ntalampiras, I. Potamitis, and N. Fakotakis. On acoustic surveillance of hazardous situations. In *ICASSP*, 2009. 4.5

[55] H. Okuno, T. Ogata, and K. Komatani. Computational auditory scene analysis and its application to robot audition: Five years experience. In *Second International Conference on Informatics Research*, 2007. 4.5

[56] C. Wooters and M. Huijbregts. *The ICSI RT07s Speaker Diarization System*. Springer-Verlag, 2008. 4.5

[57] Gal Elidan and Nir Friedman. Learning the dimensionality of hidden variables. In *Proc. UAI*, 2001. 4.6

[58] Matthew J. Beal. Variational Algorithms for Approximate Bayesian Inference. PhD Thesis, Gatsby Computational Neuroscience Unit, University College London., 2002. 4.6, 7.1.1

[59] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. 5.1

[60] R. Horst and P. M. Pardalos, editors. *Handbook of Global Optimization*. Kluwer, 1995. 5.1

[61] S. Soatto, G. Doretto, and Y. Wu. Dynamic Textures. *Intl. Conf. on Computer Vision*, 2001. 5.1

[62] E. Keogh and T. Folias. The UCR Time Series Data Mining Archive, 2002. 5.1

[63] Seth L. Lacy and Dennis S. Bernstein. Subspace identification with guaranteed stability using constrained optimization. *IEEE Transactions on Automatic Control*, 48(7):1259–1263, July 2003. 5.1, 5.2, 5.4

[64] N. L. C. Chui and J. M. Maciejowski. Realization of stable models with subspace methods. *Automatica*, 32(100):1587–1595, 1996. 5.2

[65] T. Van Gestel, J. A. K. Suykens, P. Van Dooren, and B. De Moor. Identification of stable models in subspace identification by using regularization. *IEEE Transactions on Automatic Control*, 49(9):1416–1420, 2001. 5.2

[66] Andrew Y. Ng and H. Jin Kim. Stable adaptive control with online learning. In *Proc. NIPS*, 2004. 5.3.2

[67] M. Wagner. A national retail data monitor for public health surveillance. *Morbidity and Mortality Weekly Report*, 53:40–42, 2004. 5.4.3

[68] Michael Littman, Richard Sutton, and Satinder Singh. Predictive representations of state. In *Advances in Neural Information Processing Systems (NIPS)*, 2002. 6.1, 6.4.1

[69] Herbert Jaeger. Observable operator models for discrete stochastic time series. *Neural Computation*, 12:1371–1398, 2000. 6.1, 6.1.2, 6.2, 6.2.1, 6.2.2, 6.2.5, 6.4.1, A.5

[70] Vijay Balasubramanian. Equivalence and Reduction of Hidden Markov Models. MSc. Thesis, MIT, 1993. 6.1, 6.2, 6.4.1

[71] M. P. Schützenberger. On the definition of a family of automata. *Inf Control*, 4:245–270, 1961. 6.1, 6.2, 6.4.1

[72] M. Fleiss. Matrices deHankel. *J. Math. Pures Appl.*, 53:197–222, 1974. 6.1, 6.2, 6.4.1

[73] Byron Boots, Sajid M. Siddiqi, and Geoffrey J. Gordon. Closing the Learning-Planning Loop with Predictive State Representations. *http://arxiv.org/abs/0912.2385*, 2009. 6.1, 6.5, 7.1.3, A.5

[74] Matthew Rosencrantz and Geoffrey J. Gordon. Learning low dimensional predictive representations. In *Proc. ICML*, 2004. 6.1.1, 6.4.1, 6.5

[75] Michael James and Satinder Singh. Learning and discovery predictive state representations in dynamical systems with reset. In *Proc. ICML*, 2004. 6.1.2, 6.2.2, 6.4.1

[76] Patrick O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004. 6.2

[77] Satinder Singh, Michael James, and Matthew Rudary. Predictive state representations: A new theory for modeling dynamical systems. In *Proc. UAI*, 2004. 6.2, 6.2.1, 6.2.5, 6.4.1

[78] Sajid Siddiqi, Byron Boots, and Geoffrey J. Gordon. A constraint generation approach to learning stable linear dynamical systems. In *Proc. NIPS*, 2007. 6.3.2

[79] David Wingate and Satinder Singh. Exponential family predictive representations of state. In *Proc. NIPS*, 2007. 6.4.1

[80] Ming-Jie Zhao and Herbert Jaeger and Michael Thon. A Bound on Modeling Error in Observable Operator Models and an Associated Learning Algorithm. *Neural Computation*. 6.4.1

[81] Satinder Singh, Michael L. Littman, Nicholas K. Jong, David Pardoe, and Peter Stone. Learning predictive state representations. In *Proc. ICML*, 2003. 6.4.1

[82] Britton Wolfe, Michael James, and Satinder Singh. Learning predictive state representations in dynamical systems without reset. In *Proc. ICML*, 2005. 6.4.1

[83] Matthew Rudary and Satinder Singh. A nonlinear predictive state representation. In *Proc. NIPS*, 2003. 6.4.1

[84] Nagendra Kumar and Andreas G. Andreou. Heteroscedastic discriminant analysis and reduced rank hmms for improved speech recognition. *Journal of Speech Communication*, 26:283–297, 1998. 6.4.2

[85] Zoubin Ghahramani and Geoffrey E. Hinton. Variational learning for switching state-space models. *Neural Computation*, 12(4), 2000. 6.4.2

[86] G. A. Ackerson and K. S. Fu. On state estimation in switching environments. *IEEE Transactions on Automatic Control*, 15(1):10–17, January 1970. 6.4.2

[87] R. H. Shumway and D. S. Stoffer. Dynamic linear models with switching. *J. Amer. Stat. Assoc.*, 86:763–769, 1993. 6.4.2

[88] Y. Bar-Shalom and X. R. Li. *Estimation and Tracking*. Artech House, 1993. 6.4.2

[89] A. M. Fraser and A. Dimitriadis. Forecasting probability densities by using hidden markov models with mixed states, 1993. 6.4.2

[90] R. Chen and J. Liu. Mixture kalman filters. *Journal of the Royal Statistical Society B*, 62:493–508, 2000. 6.4.2

[91] J. M. Wang, D. J. Fleet, and A. Hertzmann. Gaussian process dynamical models. In *Proc. NIPS*, 2005. 6.4.2

[92] John Langford and Ruslan Salakhutdinov and Tong Zhang. Learning Nonlinear Dynamic Models. In *ICML*, 2009. 6.4.2

[93] E. Mossel and S. Roch. Learning nonsingular phylogenies and hidden Markov models. *Annals of Applied Probability*, 2:583–614, 2006. 6.5

[94] P. Felzenszwalb, D. Huttenlocher, and J. Kleinberg. Fast Algorithms for Large State Space HMMs with Applications to Web Usage Analysis. In *Advances in Neural Information Processing Systems (NIPS)*, volume 16, 2003. 7.1.1

[95] Joseph Gonzalez, Yucheng Low, and Carlos Guestrin. Residual splash for optimally parallelizing belief propagation. In *Proc. AISTATS*, 2009. 7.1.1

[96] J. P. Shaffer. Multiple Hypothesis Testing. *Annual Review of Psychology*, 46:561–584, 1995. 7.1.1

[97] Eric Wiewiora. Modeling Probability Distributions with Predictive State Representations. PhD. Thesis, University of California at San Diego, 2007. 1

[98] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *Proc. IJCAI*, 2003. 7.1.3

[99] Matthijs T. J. Spaan and Nikos Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220, 2005. 7.1.3

[100] Masoumeh T. Izadi and Doina Precup. Point-based Planning for Predictive State Representations. In *Proc. Canadian AI*, 2008. 7.1.3

[101] L. Song, J. Huang, A. Smola, and K. Fukumizu. Hilbert space embeddings of conditional distributions. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2009. 7.1.5

[102] Nicholas Roy, Geoffrey Gordon, and Sebastian Thrun. Finding Approximate POMDP Solutions Through Belief Compression. *JAIR*, 23:1–40, 2005. 7.1.7

[103] Michael Collins, Sanjoy Dasgupta, and Robert Schapire. A generalization of principal component analysis to the exponential family. In *Proc. NIPS*, 2001. 7.1.7

[104] Geoffrey J. Gordon. Generalized^2 Linear^2 Models. In *Proc. NIPS*, 2002. 7.1.7

[105] G. W. Stewart and Ji-Guang Sun. *Matrix Perturbation Theory*. Academic Press, 1990. A.1.2, 12, 13, 16

[106] Per-Åke Wedin. Perturbation Bounds in Connection with Singular Value Decomposition. *BIT Numer. Math.*, 12:99–111, 1972. A.1.2, 16

[107] G. W. Stewart. *Matrix Algorithms Vol 1: Basic Decompositions*. SIAM, 1998. 14, 15

[108] Colin McDiarmid. On the method of bounded differences. *Surveys in Combinatorics*, pages 148–188, 1989. A.1.3

[109] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall, 1986. A.1.5