INNOVATIONS OF HIDDEN MARKOV MODELS

Xiaoyuan Ma

A Dissertation submitted to the Graduate Faculty
of the University of Virginia in Candidacy for the Degree of
Doctor of Philosophy in Statistics

Department of Statistics

University of Virginia

May 2023

Jordan Rodu, Chair

Negin Alemazkoor

Tianxi Li

Shan Yu

Innovations of Hidden Markov Models

Xiaoyuan Ma

(ABSTRACT)

Hidden Markov Models (HMM) are widely used to represent time series with regime switching phenomenon. The most popular inference method of HMM is the Baum-Welch (B-W) algorithm, which is a special case of the E-M algorithm. However, it can be too slow for many real-time applications and is prone to being trapped into local optima. Spectral learning of HMM (SHMM), which is based on the method of moments estimation, has been proposed in the literature to overcome these obstacles. Despite its promises, asymptotic theory for SHMM has been elusive, and the long-run performance of SHMM can degrade due to unchecked propagation of error. In this thesis, for spectral estimation, we studied the theoretical property of the approximation error, improved the spectral learning by adding a 'project-onto-simplex' regularization, and provided the online learning of SHMM. We compare the performance of the proposed method with the state-of-the-art methods on both simulated data and real world applications, and find that the new method not only retains the computational advantages of SHMM, but also provides more robust estimation and forecasting.

SHMM is dedicated to the endogenous evolution of HMM. Some HMM's evolution is impacted by cross predictors. We propose a variant of HMM called latent control HMM, which provides a binary latent variable controlling whether each cross predictor has impact on the transition or not. This model is inferred by standard Markov

Chain Monte Carlo (MCMC) sampling, which provides the predictability and interpretability. We show the latent control HMM's performance on both simulated and real-world data sets.

Keywords: Hidden Markov models; Spectral estimation; Method of moments; Projection-onto-simplex; Online learning; Bayesian variable selection; Bayesian hidden Markov models; Financial time series forecasting; Disease progression modeling

Dedication

This is for my family.

Acknowledgments

I would like to express my sincere gratitude to my PhD adviser, Prof. Jordan Rodu, who always supports me and guides me on the right track. I want to thank him for his brilliant ideas and kind attitude. Under his supervision, I had a happy PhD study experience.

I want to thank my committee members, Prof. Negin Alemazkoor, Prof. Tianxi Li, and Prof. Shan Yu. Prof. Alemazkoor provided me with the funding for one year and half, and we had a pleasant collaboration on gastric cancer research. I love Prof. Li's classes and lectures, which inspired me a lot. All of them provided insightful advice and suggestions during my dissertation preparations.

I also want to thank Prof. Daniel M. Keenan and Prof. Karen Kafadar, who encouraged me a lot, especially in my PhD application process and the early years of my PhD studies. In addition, I want to thank all members of the department of Statistics.

Lastly, I would like to thank my parents. They gave me unlimited love and emotional support in my life. They always respect my choices and my thoughts. Without their support, I cannot have such a wonderful journey at University of Virginia.

Contents

Chapter 1

Introduction

The Hidden Markov Model (HMM) (Baum and Petrie, 1966; Baum and Eagon, 1967) is a stochastic probabilistic model for sequential or time series data assuming that the observations are dependent on an underlying Markov chain, which is frequently used in a wide range of areas such as finance (Mamon and Elliott, 1995; Bhar and Hamori, 2004), natural language processing (Viterbi, 1967) and biology (Needle and Wunsch, 1970; Sankoff, 1972). In this dissertation, we will study two categories of HMM. One is standard HMM only involving endogenously, where we improved its spectral learning in the (1) theory, (2) methodology and (3) online learning variants to bridge the usability gap. The other category is HMM with exogenous input, where we proposed its variant latent control HMM for better handling sparsity of coefficients.

The most popular inference method for a standard HMM is the Baum-Welch (B-W) algorithm (Baum, Petrie, et al., 1970), which is a special case of the Expectation-Maximization (E-M) algorithm (Dempster, Laird, and Rubin, 1977) based on Maximum Likelihood Estimation (MLE). However, the E-M algorithm can require a large number of iterations until the parameter estimation converges, which has a large computational cost especially for large-scale time series data and might get trapped in local optima. In order to overcome these issues especially for large and high dimensional time series, Hsu, Kakade, and Zhang (2012) proposed a Spectral learning algorithm for HMM (SHMM), which is based on the Method Of Moments (MOM)

estimation and has attractive theoretical properties. However, the asymptotic behavior of the algorithm was not well-characterized. Later, Rodu (2014) improved the spectral estimation algorithm and extended it to HMMs with high dimensional and continuously distributed output, but again did not address the asymptotic properties. In this manuscript, we provide a theoretical discussion of the asymptotic distribution of SHMM algorithms. This is the first project in this thesis.

In addition to investigating the asymptotic behavior, we provide a novel improvement to the SHMM family of algorithms. Our improvement is motivated from an extensive simulation study of the methods proposed by Hsu, Kakade, and Zhang (2012) and Rodu (2014), where we found that spectral estimation does not provide stable results under the low signal-noise ratio setting. We found that the issue comes from the unchecked propagation error during the recursive forecasting in SHMM, so we propose a new spectral estimation method, the Projected SHMM (PSHMM), which leverages a novel regularization technique that we name as 'projection-onto-simplex' regularization. The PSHMM largely retains the computational advantages of SHMM methods and substantially improved the prediction stability. We demonstrate its performance by extensive simulations and real-world applications. This is the second project in this thesis.

Besides, we also provide a novel extension of SHMM and PSHMM to allow for online learning, which has two advantages. First, it speeds up computational speed for learning a model in large data with online settings. Second, it enables incorporating "forgetfulness" so it is adapting to changing dynamics of the data. This speed and flexibility is crucial, for instance, in high frequency trading. We show the effectiveness of the PSHMM in online learning settings, which has an impressive computational speed and predictability in dynamic settings. This is the third project in this thesis.

The above spectral learning is for standard HMM without the outside signals. Some variants of HMM were proposed to provide a way to leverage exogenous information. Control HMM (see e.g. Bengio and Frasconi, 1994) provides a framework for involving the cross features into HMM's transitions among hidden states inferred by MLE. Shirley et al. (2010) also proposed a model where the transition matrix is controlled by the exogenous predictors in a multinomial-logistic form under Bayesian framework and inferred by Markov Chain Monte Carlo (MCMC) sampling (Andrieu et al., 2003). However, these models work only when we are sure which exogenous features have impact on the transitions because they cannot handle sparsity well. To deal with sparsity, we proposed the latent control HMM which has a hierarchical structure where a set of binary latent control variables controlling whether each cross predictor has impact on the transition or not, by using the spike-and-slab priors (Ishwaran and Rao, 2005b). This is a Bayesian model inferred by MCMC sampling, which has two benefits. The first benefit is the predictability when the exogenous features only have a weak relationship with the transitions, because there are MCMC samples on the binary latent control variables indicating whether each feature is in the model or out of the model, which is essentially a Bayesian model averaging. Second, the MCMC samples of latent control variables can provide an interpretation that the posterior probability of each exogenous predictor is influencing the HMM transitions. This model could be applied in low frequency data, for example, low frequency trading prediction and clinical trial time series. These data could be modeled by HMM whose transitions are impacted by exogenous features. This is the fourth project in this thesis.

The structure of this dissertation is as follows: In Chapter 2 we provide theorems on the finite sample properties of SHMM for both discrete and continuous output HMM

along with a literature review for HMM, B-W algorithm and SHMM. PSHMM is introduced in Chapter 3 and the online learning variant for both SHMM and PSHMM is introduced in Chapter 4, and there are comprehensive simulations and real-world applications provided in these chapters. In Chapter 5 we discussed deeply about the latent control HMM, along with the simulations and real-world data applications with their comparisons with the state-of-the-art methods. Chapter A contains more technical and results details.

Chapter 2

Theory of SHMM

In this chapter, we study the asymptotic behavior of SHMM. Before that, we formally introduce HMM and its inference algorithm including Baum-Welch algorithm (Baum and Eagon, 1967) and SHMM (Hsu, Kakade, and Zhang, 2012; Rodu et al., 2013).

2.1 Literature Review

2.1.1 Hidden Markov models

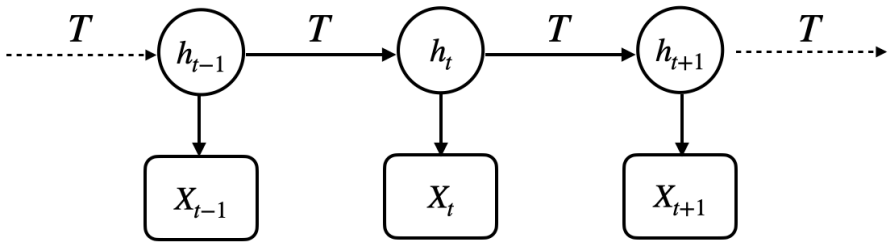


Figure 2.1: Model structure of standard HMM. $\{h_t\}$ is a latent Markov chain evolves according to transition matrix $\mathbf{T}$. For each time stamp t, the observed X_t is generated according to the emission distribution associated with h_t.

The standard HMM (Baum and Petrie, 1966; Baum and Eagon, 1967; Baum, Petrie, et al., 1970) is defined that we have a latent state process $\{h_t\}_t$ that evolves between S hidden categorical states $\{1, 2, \cdots, S\}$ according to a Markov chain, where h_t is the

hidden state at time t. The Markov chain is characterized by an initial probability $\pi_0 = [\pi_0^{(1)}, \cdots, \pi_0^{(S)}]$ where $h_1 \sim Categorical(\pi_0^{(1)}, \cdots, \pi_0^{(S)})$ and a transition matrix $\mathbf{T} = [\mathbf{T}_{ij}]_{i=1,\cdots,S}^{j=1,\cdots,S}$ where $\mathbf{T}_{ij} = \mathrm{P}(h_{t+1} = j | h_t = i)$ for $\forall t$. The emitted observation X_t is distributed when the value of the hidden state at time t, $X_t | h_t = s \sim \mathcal{F}_s$ where $\mathcal{F}_s$ is the emission distribution conditioned on the hidden state $h_t = s$. The standard HMM's parameters are $\left(\pi_0, \mathbf{T}, \{\mathcal{F}_s\}_{s=1}^S\right)$, and we denote $\{\mathcal{F}_s\}_{s=1}^S$ as $\mathcal{E}$. Figure 2.1 is a graphical representation of the standard HMM. For continuous output HMM h_t denotes the hidden cluster of observations of X_t. Typically, if the emission follows a Gaussian distribution, then we call it Gaussian HMM (GHMM), which is similar to Gaussian Mixture Models (GMM) (McLachlan and Basford, 1988) but with temporal dependency.

2.1.2 Standard inference method for HMM: Baum-Welch algorithm

The Baum-Welch algorithm (Baum and Petrie, 1966) is a variant of E-M algorithm (Dempster, Laird, and Rubin, 1977) used for inferring the unobserved hidden Markov chain $\{h_t\}_t$ and parameters $(\pi_0, \mathbf{T}, \mathcal{E})$ by MLE. Predicting the next observation can be done with the forward propagation algorithm (Baum and Petrie, 1966). Specifically, in the Baum-Welch algorithm, there are two iterative main steps. In the E-step, the Baum-Welch algorithm calculates the forward propagation probability $\alpha_t(j), j = 1, \cdots, S$ and backward propagation probability $\beta_t(j), j = 1, \cdots, S$. Then we can calculate state occupancy probabilities $\mathrm{P}(h_t = j)$ and $\xi_t(i, j) = \mathrm{P}(h_t = i, h_{t+1} = j)$. In the M-step, the Baum-Welch algorithm updates parameters $\{\widehat{\pi}_0, \widehat{\boldsymbol{T}}, \widehat{\mathcal{E}}\}$ conditioned on the state occupancy probabilities. See Algorithm 1 for more details.

Algorithm 1: Baum-Welch Algorithm for HMM

Data: $\{X_t\}_{t=1}^{T}$

Result: $\{\widehat{\pi}_0, \widehat{T}, \widehat{\mathcal{E}}\}$

Set $\{\widehat{\pi}_0, \widehat{T}, \widehat{\mathcal{E}}\}$ with random initial conditions;

while *not converged* **do**

`// E-step: update` $\{\gamma_t(j)\}_{t=1:T}^{j=1:S}$ `and` $\{\xi_t(i,j)\}_{t=1:T}^{i,j=1:S}$ `conditioned on` $\{\widehat{\pi}_0, \widehat{T}, \widehat{\mathcal{E}}\}$

Find $\{b_j(X_t)\}_{t=1:T}^{j=1:S}$;

for $t \leftarrow 1$ **to** T **do**

$$\alpha_t(j) \leftarrow \begin{cases} \pi_0(j) b_j(X_1), & \text{if } t = 1 \\ \sum_{i=1}^{S} \alpha_{t-1}(i) T_{ij} b_j(X_t), & \text{otherwise} \end{cases} \quad \forall j = 1, \cdots, S;$$

end

for $t \leftarrow T$ **to** 1 **do**

$$\beta_t(j) \leftarrow \begin{cases} 1, & \text{if } t = T \\ \sum_{j=1}^{S} T_{ij} \beta_{t+1}(j) b_j(X_{t+1}), & \text{otherwise} \end{cases} \quad \forall j = 1, \cdots, S;$$

end

$\mathrm{P}(h_t = j) = \gamma_t(j) \leftarrow \frac{\alpha_t(j)\beta_t(j)}{\sum_{i=1}^{S} \alpha_t(i)\beta_t(i)}$ $\forall t, j$;

$\mathrm{P}(h_t = i, h_{t+1} = j) = \xi_t(i,j) \leftarrow \frac{\alpha_t(i) T_{ij} b_j(X_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{S} \alpha_t(i)\beta_t(i)}$ $\forall t, i, j$;

`// M-step: update` $\{\widehat{\pi}_0, \widehat{T}, \widehat{\mathcal{E}}\}$ `conditioned on` $\{\gamma_t(j)\}$ `and` $\{\xi_t(i,j)\}$

$\widehat{\pi}_0(j) \leftarrow \gamma_1(j)$ $\forall j = 1, \cdots, S$;

$\widehat{T}_{ij} \leftarrow \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{k=1}^{S} \xi_t(i,k)}$ $\forall i, j = 1, \cdots, S$;

Estimate $\widehat{\mathcal{E}}_j$ based on weighted samples $\{X_t\}_{t=1}^{T}$ with weights $\{\gamma_t(j)\}_{t=1}^{T}$;

end

Strengths of Baum-Welch algorithm The inference based on the Baum-Welch algorithm is very intuitive since it infers the hidden parameters based on forward-backward propagation under E-M framework, which is the reverse of the data generation process. It obtains an unbiased estimation with achieving the global optima theoretically (Yang, Balakrishnan, and Wainwright, 2017). It is robust in parameter estimation and prediction, because it estimates parameters and predicts based on the state occupancy probabilities of hidden states which are regularized to be non-zero and reasonable. For prediction, the Baum-Welch algorithm first predicts the weights,

which are explicitly guaranteed to be non-negative and normalized to sum-up 1, and then yields the prediction as a weighted combination of cluster means that $\hat{y}_t = M\hat{w}_t$. This is later the motivation of proposing PSHMM in Chapter 3.

Limitations of Baum-Welch algorithm While the E-M algorithm is widely used to infer HMMs, there are some limitations. First of all, the Baum-Welch algorithm is a special case of the E-M algorithm, which infers HMMs based on MLE. The E-M algorithm usually needs many iterations to achieve convergence and is computationally expensive. In addition, since the likelihood of HMMs are not convex, the E-M algorithm might be trapped into local optima (Chen, 2022). Although this could be mitigated by trying different initialization settings, it is hard to avoid completely. Also for high dimensional data sets, the E-M algorithm might be stuck due to accumulated truncation error (Jamshidian and Jennrich, 2000).

2.1.3 Estimating HMM by spectral learning

Spectral estimation is to estimate models based on MOM estimation (Pearson, 1936). MOM estimators usually don't require iterations, so in general they will be much faster than the E-M algorithm. Spectral estimation is a well-studied topic that people have developed spectral estimations for many statistical machine learning methods. For example, Hsu and Kakade (2013) studied spectral learning for GMM. Anima Anandkumar et al. (2012) developed the spectral estimation for latent Dirichlet allocation. Animashree Anandkumar et al. (2011) proposed spectral learning for tree methods.

For HMMs, Jaeger (2000) was the first person who found that HMM could be rewritten into the multiplications of some operators. One decade later, Hsu, Kakade, and

Zhang (2012) proposed SHMM, a noval framework for learning HMM by spectral estimation. Since HMM inferred by Baum-Welch algorithm has a high computational complexity, this new framework for inferring discrete output HMMs and its great performance especially its low computational cost bring itself into notice. Later on, Rodu (2014) further improved this method by reducing the data to a low dimensional subspace and extended the spectral learning framework to continuous output HMMs. Rodu et al. (2013) also raised a new way to build the moment estimations by using regression models. Both Hsu, Kakade, and Zhang (2012) and Rodu (2014) provided nice mathematical properties of SHMM.

2.1.4 Spectral estimation HMMs by Hsu, Kakade, and Zhang (2012)

In this part we briefly introduced the spectral estimation method proposed by Hsu, Kakade, and Zhang (2012). Recall h_t denotes the hidden state at time t, and X_t the emitted observation. Hsu, Kakade, and Zhang (2012) wrote the likelihood with the observable operators as below:

$$Pr(x_{1:t}) = b_\infty^T B(x_t)B(x_{t-1})\cdots B(x_1)b_1 \tag{2.1}$$

where,

$$\begin{aligned} b_1 &= U^T P_1, \quad b_\infty^T = P_1(U^T P_{21})^T, \quad B(x) = (U^T P_{3x1})(U^T P_{21})^T, \\ P_1 &= E\pi, \quad P_{21} = E(x_2 \otimes x_1), \quad P_{3x1} = E(x_3 \otimes x_1 \otimes x_2)x, \end{aligned}$$

and $\otimes$ represents the outer product of two vectors.

Therefore, we can approximate the likelihood by plugging the method of moments estimation of P_1, P_{21} and P_{3x1} that

$$\hat{Pr}(x_{1:t}) = \hat{b}_\infty^T \hat{B}(x_t)\hat{B}(x_{t-1})\cdots\hat{B}(x_1)\hat{b}_1. \tag{2.2}$$

Based on this approximation, we could get the conditional predictive probability of x_t given x_{t-1} without inferring the hidden states by the following recursive formulas:

$$\begin{aligned} \hat{b}_{1:t} &= \frac{\hat{B}_{x_t}\hat{b}_t}{\hat{b}_\infty^\top \hat{B}_{x_t}\hat{b}_t} \\ \hat{Pr}(x_t|x_{1:t-1}) &= \frac{\hat{b}_\infty^\top \hat{B}_{x_t}\hat{b}_t}{\hat{b}_{\sum_x \infty}^\top \hat{B}_{x_t}\hat{b}_t}. \end{aligned} \tag{2.3}$$

This recursive formula could be used for prediction.

2.1.5 Spectral estimation HMMs by Rodu (2014)

The model proposed by Rodu (2014) is shown in Figure 2.2. For estimation of the model, different from Hsu, Kakade, and Zhang (2012), theis method further reduces observations onto a lower dimensional space of dimensionality d and these reduced observations are denoted as $y_t = U^\top x_t$. Here U could be obtained from singular value decomposition (SVD) (Eckart and Young, 1936) on the bigrams $E(X_2 \otimes X_1)$. In this dissertation, we use y_t to denote the observation with reduced dimensionaliy, and use $\hat{y}_t$ to denote its prediction conditioned on the information up to $t-1$.

We discuss the choice of dimensionality d and the projection of the observations in Section 3.3. Figure 2.3 shows how to make prediction based on the recursive predictions formula by Equation 2.6.

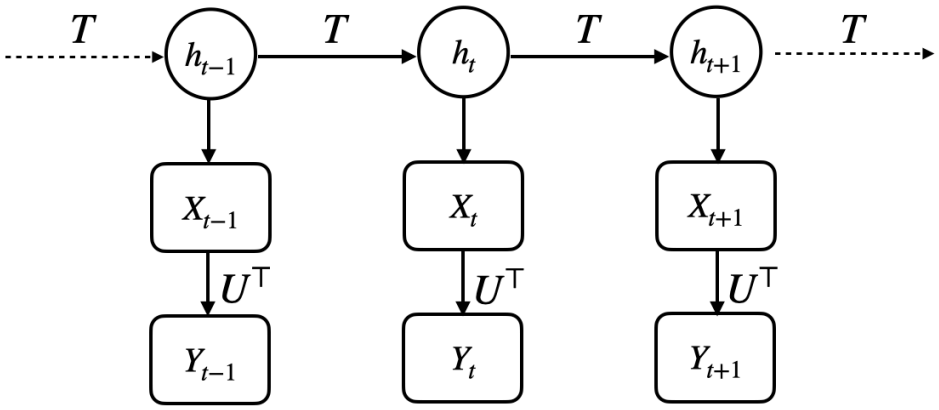


Figure 2.2: Spectral estimation model by Rodu (2014). Besides the latent state series $\{h_t\}_t$ and observed series $\{X_t\}_t$, Rodu (2014) introduced a reduced-dimensional series $\{Y_t = U^\top X_t\}$ which is a projection of X_t on a lower-dimensional subspace whose dimensionality is equal to the number of hidden states. Then the spectral estimation will be performed based on $\{Y_t\}_t$.

Using the spectral estimation framework, the likelihood can be written as:

$$Pr(x_{1:t}) \quad = \quad c_\infty^\top C(y_t)C(y_{t-1})\cdots C(y_1)c_1, \tag{2.4}$$

where

$$c_1 = \mu, c_\infty^\top = \mu^\top \Sigma^{-1}, C(y) = K(y)\Sigma^{-1},$$

$$\mu = E(y_1) = U^\top M\pi,$$

$$\Sigma = E(y_2 y_1^\top) = U^\top MTdiag(\pi)M^\top U,$$

$$K(a) = \mathrm{E}(y_3 y_1^\top y_2^\top)a = U^\top MTdiag(M^\top Ua)Tdiag(\pi)(M^\top U),$$

$$M = [M_1, \cdots, M_S] \text{ where } M_i = \mathrm{E}(X|i).$$

These quantities can be empirically estimated as:

$$\widehat{Pr}(x_{1:t}) \quad = \quad \hat{c}_\infty^\top \hat{C}(y_t)\hat{C}(y_{t-1})\cdots\hat{C}(y_1)\hat{c}_1, \tag{2.5}$$

where

$$\hat{c}_1 = \hat{\mu}, \hat{c}_\infty^\top = \hat{\mu}^\top \hat{\Sigma}^{-1}, \hat{C}(y) = \hat{K}(y)\hat{\Sigma}^{-1},$$
$$\hat{\mu} = \tfrac{1}{N}\textstyle\sum_{i=1}^{N} Y_i,$$
$$\hat{\Sigma} = \tfrac{1}{N}\textstyle\sum_{i=1}^{N-1} Y_{i+1}Y_i^\top,$$
$$\hat{K}(y) = \tfrac{1}{N}\textstyle\sum_{i=1}^{N-2} Y_{i+2}Y_i^\top \cdot Y_{i+1}^\top y.$$

Prediction of y_t is computed recursively by:

$$\hat{y}_t \quad = \quad \frac{C(y_{t-1})\hat{y}_{t-1}}{c_\infty^\top C(y_{t-1})\hat{y}_{t-1}}. \tag{2.6}$$

Note that the recursive prediction by Equation 2.6 only depends on the estimated prediction $\hat{y}_{t-1}$ at the last time stamp $t-2$, and the observation y_{t-1} of this time stamp. Predicted x_t can be recovered as:

$$\hat{x_t}|x_1, x_2, \cdots, x_{t-1} = U\hat{y}_t. \tag{2.7}$$

The above exposition of spectral likelihood estimation assumes a discrete output HMM. For a continuous output HMM, the spectral estimation of likelihood is slightly different. We need some kernel function $G(x)$ to calculate K, so

$$K(a) = U^\top MTdiag(M^\top UG(a))Tdiag(\pi)(M^\top U).$$

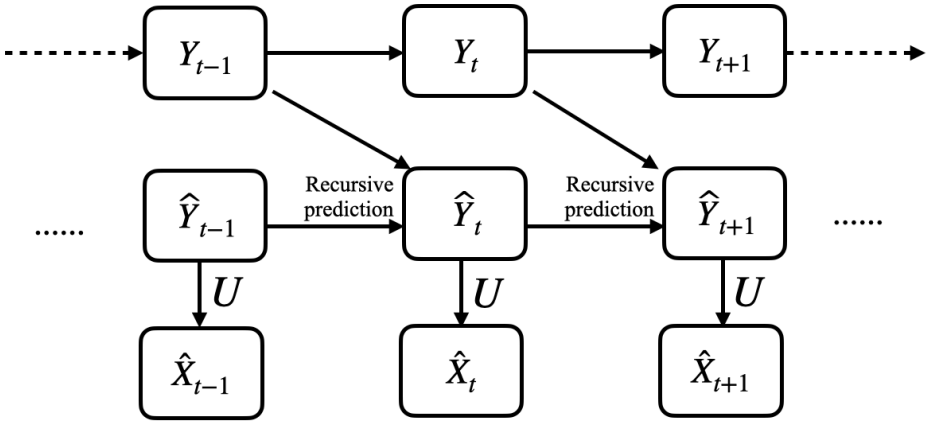


Figure 2.3: Prediction of spectral estimation model by Rodu (2014). The recursive prediction is by Equation 2.6.

(For more on $G(\cdot)$ see Rodu, 2014). In this paper, we will use a linear kernel $G(a) = a$ for simplicity. In this case, the moment estimation and recursive forecasting for the continuous case are identical to the discrete case. See Rodu et al. (2013) for detailed derivations and mathematical proofs of these results. Rodu (2014) also studied the theoretical properties of SHMM and also proved that the likelihood observable operator representation (i.e. spectral estimation of likelihood) will converge to the underlying truth almost surely.

We have introduced both the spectral learning algorithms proposed by Hsu, Kakade, and Zhang (2012) and Rodu (2014). We want to emphasize that the later one is much faster in the computational aspect in high-dimensional cases, because we first reduce the observations on a reduced-rank subspace. Our study on spectral estimation will be based on this method.

2.2 Theoretical Properties of SHMM

In general, SHMM estimates the likelihood through MOM estimators. Although MOM gives fast approximation, the theoretical properties of SHMM estimation are less well studied. Hsu, Kakade, and Zhang, 2012 and Rodu et al., 2013 give the conditions where the spectral estimator converges to the true likelihood almost surely:

$$\hat{Pr}(x_{1:t}) = \hat{b}_\infty^T \hat{B}(x_t)\hat{B}(x_{t-1})\cdots\hat{B}(x_1)\hat{b}_1 \xrightarrow{a.s.} Pr(x_{1:T}), \tag{2.8}$$

$$\widehat{Pr}(x_{1:T}) = \hat{c}_\infty^\top \hat{C}(y_t)\hat{C}(y_{t-1})\cdots\hat{C}(y_1)\hat{c}_1 \xrightarrow{a.s.} Pr(x_{1:T}), \tag{2.9}$$

as $N \to \infty$ where N is the number of training triples will be discussed later.

In this manuscript, we study the asymptotic distribution of $\hat{Pr}(x_{1:T}) - Pr(x_{1:T})$. Theorem 2.2 shows a Central Limit Theorem (CLT)-type bound of the approximation error. Note that in the following discussions on theoretical properties, we use the same setting as Hsu, Kakade, and Zhang (2012) and Rodu (2014) above that the $x_{1:T}$ is a fixed target process that we will compute the likelihood on. Also there is another training set independent from the target process, but is generated from the same HMM as the target. This training set contains N i.i.d. triples for estimating μ, Σ and K. We will study the asymptotic property of estimated likelihood by SHMM as $N \to \infty$.

2.2.1 Likelihood decomposition by spectral estimation

First, however, we identify the sources of error for the SHMM in Lemma 2.1. Lemma 2.1 makes use of the 'Δ' terms, including $\widehat{\Delta\mu}$, $\widehat{\Delta\Sigma}$ and $\widehat{\Delta K}$, defined through the following equations: $\widehat{\mu} = \mu + \widehat{\Delta\mu}$, $\widehat{\Sigma} = \Sigma + \widehat{\Delta\Sigma}$, $\widehat{K} = K + \widehat{\Delta K}$.

Lemma 2.1.

$$\widehat{Pr}(x_{1:T}) \;=\; Pr(x_{1:T}) + (v+\tilde{v})^\top \widehat{\Delta\mu} + \sum_{t=1}^{T} a_t^\top \widehat{\Delta K}(y_t)\tilde{a_t} - \sum_{t=0}^{T} b_t^\top \widehat{\Delta\Sigma}\tilde{b_t} + \mathcal{O}_p(N^{-1}),$$

where

$$\begin{aligned}
v &= \left(\mu^\top \Sigma^{-1} K(y_T) \cdots K(y_1)\Sigma^{-1}\right)^\top; & \tilde{v} &= \Sigma^{-1}K(y_T)\cdots K(y_1)\Sigma^{-1}\mu;\\
a_t &= \left(\mu^\top \Sigma^{-1} K(y_T)\Sigma^{-1} \cdots K(y_{t+1})\Sigma^{-1}\right)^\top; & \tilde{a_t} &= \Sigma^{-1}K(y_{t-1})\cdots K(y_1)\Sigma^{-1}\mu;\\
b_t &= \left(\mu^\top \Sigma^{-1} K(y_T)\Sigma^{-1} \cdots \Sigma^{-1}K(y_{t+1})\Sigma^{-1}\right)^\top; & \tilde{b_t} &= \Sigma^{-1}K(y_t)\Sigma^{-1}\cdots K(y_1)\Sigma^{-1}\mu.
\end{aligned}$$

Proof (Lemma 2.1)**.** The basic strategy is to fully expand $\widehat{Pr}(x_{1:T})$ after rewriting the estimated quantities as a sum of the true quantity plus an error term. We then categorize each summand based on how many 'Δ' terms it has. There are three categories: terms with zero 'Δ' terms (i.e. the true likelihood $Pr(x_{1:T})$), terms with only one 'Δ' (i.e. $(v+\tilde{v})^\top \widehat{\Delta\mu} + \sum_{t=1}^{T} a_t^\top \widehat{\Delta K}(y_t)\tilde{a_t} - \sum_{t=0}^{T} b_t^\top \widehat{\Delta\Sigma}\tilde{b_t}$), and all remaining terms, which involve at least two 'Δ' quantities, and can be relegated to $\mathcal{O}_p(N^{-1})$.

First, we expand the estimated likelihood by decomposing it into the underlying truth

plus the error terms. We have

$$\begin{aligned}
& \widehat{Pr}(x_{1:T}) \\
= \; & \hat{c}_\infty^\top \hat{C}(y_T)\hat{C}(y_{T-1})\cdots\hat{C}(y_1)\hat{c}_1 \\
= \; & (\hat{\mu}^\top\hat{\Sigma}^{-1})[\hat{K}(y_T)\hat{\Sigma}^{-1}][\hat{K}(y_{T-1})\hat{\Sigma}^{-1}]\cdots[\hat{K}(y_1)\hat{\Sigma}^{-1}]\hat{\mu} \\
= \; & [(\mu+\widehat{\Delta\mu})^\top(\Sigma+\widehat{\Delta\Sigma})^{-1}][(K+\widehat{\Delta K})(y_T)(\Sigma+\widehat{\Delta\Sigma})^{-1}] \\
& \cdots[(K+\widehat{\Delta K})(y_1)(\Sigma+\widehat{\Delta\Sigma})^{-1}](\mu+\widehat{\Delta\mu}).
\end{aligned} \tag{2.10}$$

Consider the matrix perturbation $(\Sigma+\widehat{\Delta\Sigma})^{-1} = \Sigma^{-1} - \Sigma^{-1}\widehat{\Delta\Sigma}\Sigma^{-1} + O(||\widehat{\Delta\Sigma}||^2)$, Here the matrix norm $||\cdot||$ can be any norm since all matrix norms have equivalent orders (Li, 2006). Also note that all items with $\widehat{\Delta}$ are $O_p(N^{-\frac{1}{2}})$. N is the number of *i.i.d.* triple (Y_1, Y_2, Y_3) for estimating $\widehat{\mu}, \widehat{\Sigma}, \widehat{K}$. Note that N and T are not related, and that we work in the regime where T is fixed but $N \to \infty$. For example, $\sqrt{N}\widehat{\Delta\mu} = \sqrt{N}(\hat{\mu}-\mu) = \sqrt{N}(\frac{1}{N}\Sigma_{i=1}^N Y_i - \mu) \xrightarrow{d} MVN(0, Cov(Y))$. Similar analyses apply to $\widehat{\Sigma}$ and $\widehat{K}$ can be similarly. So

$$\begin{aligned}
\widehat{\Delta\mu} &= [O_p(N^{-1/2})]^{(d)}; \\
\widehat{\Delta\Sigma} &= [O_p(N^{-1/2})]^{(d\times d)}; \\
\widehat{\Delta K} &= [O_p(N^{-1/2})]^{(d\times d\times d)},
\end{aligned}$$

where d is the dimension of Y_t. One application is $(\Sigma+\widehat{\Delta\Sigma})^{-1} = \Sigma^{-1} - \Sigma^{-1}\widehat{\Delta\Sigma}\Sigma^{-1} + O_p(N^{-1})$.

According to the previous two expansions, we could rewrite Equation 2.10. We could

distribute

$$
\begin{aligned}
& [(\mu+\widehat{\Delta\mu})^\top(\Sigma+\widehat{\Delta\Sigma})^{-1}][(K+\widehat{\Delta K})(y_T)(\Sigma+\widehat{\Delta\Sigma})^{-1}] \\
& \cdots[(K+\widehat{\Delta K})(y_1)(\Sigma+\widehat{\Delta\Sigma})^{-1}](\mu+\widehat{\Delta\mu}) \\
= \; & [(\mu+\widehat{\Delta\mu})^T(\Sigma^{-1}-\Sigma^{-1}\widehat{\Delta\Sigma}\Sigma^{-1}+O_p(N^{-1}))] \\
& \cdots[(K+\widehat{\Delta K})(y_T)(\Sigma^{-1}-\Sigma^{-1}\widehat{\Delta\Sigma}\Sigma^{-1}+O_p(N^{-1}))] \\
& \cdots[(K+\widehat{\Delta K})(y_1)(\Sigma^{-1}-\Sigma^{-1}\widehat{\Delta\Sigma}\Sigma^{-1}+O_p(N^{-1}))](\mu+\widehat{\Delta\mu}) \\
= \; & \mu^\top\Sigma^{-1}K(y_T)\Sigma^{-1}\cdots K(y_1)\Sigma^{-1}\mu+(v+\tilde{v})^\top\widehat{\Delta\mu}+\sum_{t=1}^{T}a_t^\top\widehat{\Delta K}(y_t)\tilde{a}_t \\
& -\sum_{t=0}^{T}b_t^\top\widehat{\Delta\Sigma}\tilde{b}_t+O_p(N^{-1}) \\
= \; & Pr(x_{1:T})+(v+\tilde{v})^\top\widehat{\Delta\mu}+\sum_{t=1}^{T}a_t^\top\widehat{\Delta K}(y_t)\tilde{a}_t-\sum_{t=0}^{T}b_t^\top\widehat{\Delta\Sigma}\tilde{b}_t+O_p(N^{-1})
\end{aligned}
$$

In the above, we first substitute $(\Sigma+\widehat{\Delta\Sigma})^{-1}$ by $\Sigma^{-1}-\Sigma^{-1}\widehat{\Delta\Sigma}\Sigma^{-1}+O_p(N^{-1})$. Then we distribute all multiplications. After distribution, there will be finite terms. We could categorize these forms into 3 categories according to different orders of convergence. The first category is the deterministic term without randomness. There is only one term in this category, $\mu^\top\Sigma^{-1}K(y_T)\Sigma^{-1}\cdots K(y_1)\Sigma^{-1}\mu$, which is $Pr(x_{1:T})$. The second category is the part that converges with order $O_p(N^{-1/2})$. This category involves all terms with one '$\widehat{\Delta}$' term, i.e. one $\widehat{\Delta\mu}$, $\widehat{\Delta\Sigma}$ or $\widehat{\Delta K}$. This category is $(v+\tilde{v})^\top\widehat{\Delta\mu}+\sum_{t=1}^{T}a_t^\top\widehat{\Delta K}(y_t)\tilde{a}_t-\sum_{t=0}^{T}b_t^\top\widehat{\Delta\Sigma}\tilde{b}_t$. For simplicity of subsequent theorem proof, we simplify each term in the following way: for each '$\widehat{\Delta}$' term, we denote the part in front of or behind them by $v,\tilde{v},a_t,\tilde{a}_t,b_t,\tilde{b}_t$ as defined in this lemma. The third categories are all remaining terms. These terms are converging faster than or in the order of $O_p(N^{-1})$. There are finite terms in this category, so the summation of them is still $O_p(N^{-1})$. □

2.2.2 Central limit theorem for likelihood approximation error

Lemma 2.1 shows how the estimated error propagates to the likelihood approximation. We can leverage the fact that our moment estimators have a CLT property to obtain the desired results in Theorem 2.2.

For consistency in notations, define a "flattening" operator $\mathcal{F}(\cdot)$ for both matrices and 3-way tensors. For matrix $A_{d\times d}$,

$$\mathcal{F}(A) = [A^{(1,1)}, A^{(1,2)}, \cdots, A^{(d,d)}]^\top;$$

For tensor $B_{d\times d\times d}$,

$$\mathcal{F}(B) = [B^{(1,1,1)}, B^{(1,1,2)}, \cdots, B^{(1,1,d)}, B^{(1,2,1)}, B^{(1,2,2)}, \cdots, B^{(1,2,d)}, \cdots, B^{(1,d,d)}, \cdots, B^{(d,d,d)}]^\top.$$

We now state and prove our main theorem.

Theorem 2.2.

$$\sqrt{N}(\widehat{Pr}(x_{1:T}) - Pr(x_{1:T})) \xrightarrow{d} N\left(0, \beta^\top Cov\left(\begin{bmatrix} Y_1 \\ \mathcal{F}(Y_2 \otimes Y_1) \\ \mathcal{F}(Y_3 \otimes Y_1 \otimes Y_2) \end{bmatrix}\right)\beta\right),$$

where

$$\beta = \left[(v+\tilde{v})^\top; -\left(\sum_{t=0}^{T} \mathcal{F}(b_t \otimes \tilde{b}_t)\right)^\top; \left(\sum_{t=1}^{T} \mathcal{F}(a_t \otimes \tilde{a}_t \otimes y_t)\right)^\top\right]^\top$$

and $v, \tilde{v}, a_t, \tilde{a}_t, b_t, \tilde{b}_t$ *are defined as in Lemma 2.1.*

Proof (Theorem 2.2)**.** We flatten $\widehat{\Delta\Sigma}$ and $\widehat{\Delta K}$ as

$$\begin{aligned}\mathcal{F}(\widehat{\Delta\Sigma}) &= [\widehat{\Delta\Sigma}^{(1,1)}, \widehat{\Delta\Sigma}^{(1,2)}, \cdots, \widehat{\Delta\Sigma}^{(d,d)}]^\top,\\ \mathcal{F}(\widehat{\Delta K}) &= [\widehat{\Delta K}^{(1,1,1)}, \widehat{\Delta K}^{(1,1,2)}, \cdots, \widehat{\Delta K}^{(d,d,d)}]^\top.\end{aligned}$$

Rewriting $a_t^\top \widehat{\Delta K}(y_t)\tilde{a}_t$ and $b_t^\top \widehat{\Delta\Sigma}\tilde{b}_t$ in Equation 2.10 as

$$\begin{aligned}& a_t^\top \widehat{\Delta K}(y_t)\tilde{a}_t\\ =\ & \sum_{i=1}^{d}\sum_{j=1}^{d}\sum_{k=1}^{d} a_t^{(i)}\tilde{a}_t{}^{(j)} y_t^{(k)} \widehat{\Delta K}^{(i,j,k)}\\ =\ & [a_t^{(1)}\tilde{a}_t{}^{(1)}y_t^{(1)}, a_t^{(1)}\tilde{a}_t{}^{(1)}y_t^{(2)}, \cdots, a_t^{(d)}\tilde{a}_t{}^{(d)}y_t^{(d)}]^\top \cdot \mathcal{F}(\widehat{\Delta K})\\ =\ & \mathcal{F}(a_t \otimes \tilde{a}_t \otimes y_t)^\top \cdot \mathcal{F}(\widehat{\Delta K}),\end{aligned}$$

and

$$\begin{aligned}& b_t^\top \widehat{\Delta\Sigma}\tilde{b}_t\\ =\ & \sum_{i=1}^{d}\sum_{j=1}^{d} b_t^{(i)} \widehat{\Delta\Sigma}^{(i,j)} \tilde{b}_t{}^{(j)}\\ =\ & [b_t^{(1)}\tilde{b}_t{}^{(1)}, b_t^{(1)}\tilde{b}_t{}^{(2)}, \cdots, b_t^{(1)}\tilde{b}_t{}^{(d)}, b_t^{(2)}\tilde{b}_t{}^{(1)}, \cdots, b_t^{(d)}\tilde{b}_t{}^{(d)}]^\top \cdot \mathcal{F}(\widehat{\Delta\Sigma})\\ =\ & \mathcal{F}(b_t \otimes \tilde{b}_t)^\top \cdot \mathcal{F}(\widehat{\Delta\Sigma}).\end{aligned}$$

That is to say, we have

$$\begin{aligned}a_t^\top \widehat{\Delta K}(y_t)\tilde{a}_t &= \mathcal{F}(a_t \otimes \tilde{a}_t \otimes y_t)^\top \cdot \mathcal{F}(\widehat{\Delta K}),\\ b_t^\top \widehat{\Delta\Sigma}\tilde{b}_t &= \mathcal{F}(b_t \otimes \tilde{b}_t)^\top \cdot \mathcal{F}(\widehat{\Delta\Sigma}).\end{aligned}$$

So

$$\begin{aligned}
& \widehat{Pr}(x_{1:T}) - Pr(x_{1:T}) \\
= & \left[(v+\tilde{v})^\top; -\left(\sum_{t=0}^{T} \mathcal{F}(b_t \otimes \tilde{b}_t)\right)^\top; \left(\sum_{t=1}^{T} \mathcal{F}(a_t \otimes \tilde{a}_t \otimes y_t)\right)^\top\right] \cdot \begin{bmatrix} \widehat{\Delta\mu} \\ \mathcal{F}(\widehat{\Delta\Sigma}) \\ \mathcal{F}(\widehat{\Delta K}) \end{bmatrix} \\
& + O_p(N^{-1}) \\
= & \beta^\top \cdot \widehat{\Delta\theta} + O_p(N^{-1}).
\end{aligned}$$

Since the central limit theorem applies separately to $\widehat{\Delta\mu}, \widehat{\Delta\Sigma}, \widehat{\Delta K}$, then

$$\begin{aligned}
\sqrt{N}\widehat{\Delta\theta} &= \sqrt{N} \begin{bmatrix} \frac{1}{N}\sum_{i=1}^{N} Y_{i,1} - \mu \\ \mathcal{F}(\frac{1}{N}\sum_{i=1}^{N} Y_{i,2} \otimes Y_{i,1} - \Sigma) \\ \mathcal{F}(\frac{1}{N}\sum_{i=1}^{N} Y_{i,3} \otimes Y_{i,1} \otimes Y_{i,2} - K) \end{bmatrix} \\
&\xrightarrow{d} MVN\left(\vec{0}, Cov\left(\begin{bmatrix} Y_1 \\ \mathcal{F}(Y_2 \otimes Y_1) \\ \mathcal{F}(Y_3 \otimes Y_1 \otimes Y_2) \end{bmatrix}\right)\right).
\end{aligned}$$

Therefore,

$$\sqrt{N}(\widehat{Pr}(x_{1:T}) - Pr(x_{1:T})) \xrightarrow{d} N\left(0, \beta^\top Cov\left(\begin{bmatrix} Y_1 \\ \mathcal{F}(Y_2 \otimes Y_1) \\ \mathcal{F}(Y_3 \otimes Y_1 \otimes Y_2) \end{bmatrix}\right)\beta\right).$$

□

For simplicity of representation, we derived Theorem 2.2 under the assumption that the output distribution is discrete. For continuous output, the central limit theorem

still holds with a proper kernel function $G(\cdot)$ as mentioned in section 2.1.3.

2.3 Experimental Validation of Theorem 2.2

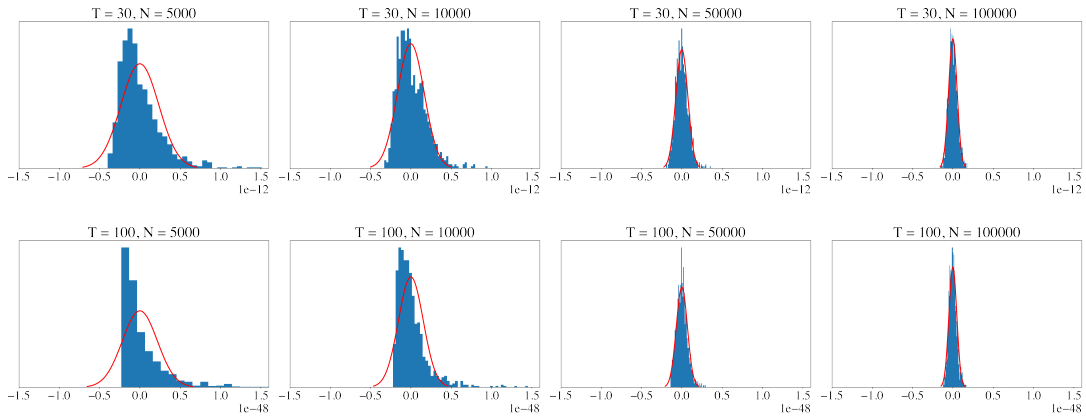
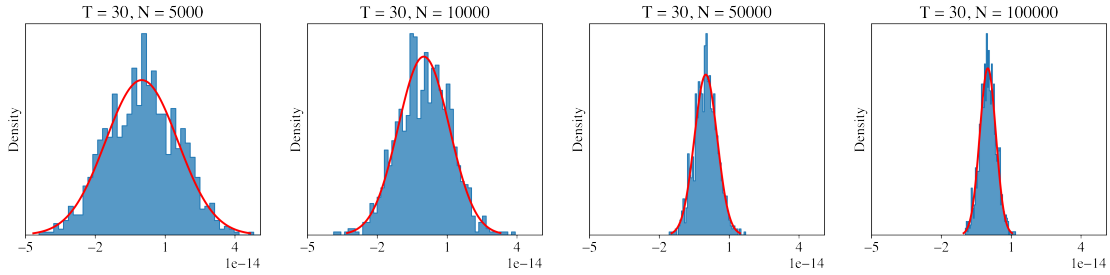


Figure 2.4: Empirical histograms of $\hat{Pr}(x_{1:T}) - Pr(x_{1:T})$ estimated under different training size N and length T and theoretical density calculated based on Theorem 2.2. Each subfigure is associated with a different N. We could see that as N goes larger, the distribution converges to the theoretical normal distribution. When T is smaller, the estimation error converges more quickly to the asymptotic distribution.

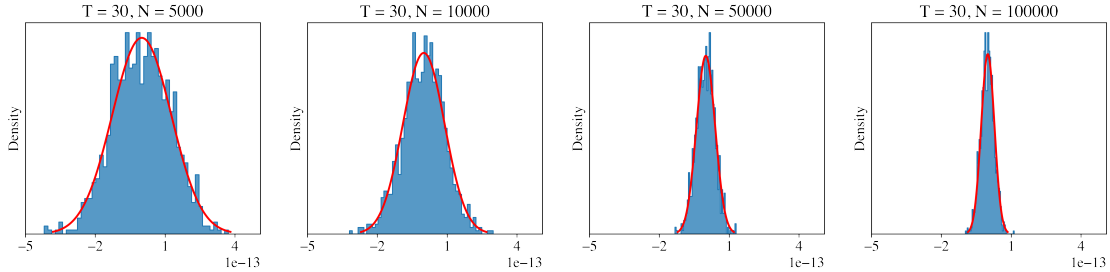
We performed a series of experiments to validate the conclusions of theorem 2.2. We generated a target series $x_{1:T}$ with $x_i \in \mathbb{R}^3$ and $T = 30, 100$ using a 3-state GHMM, where the initial probabilities and sticky transition probabilities are as described in Section 3.4, and with discrete emission probability matrix

$$[[0.8, 0.1, 0.1]^\top, [0.1, 0.8, 0.1]^\top, [0.1, 0.1, 0.8]^\top]^\top.$$

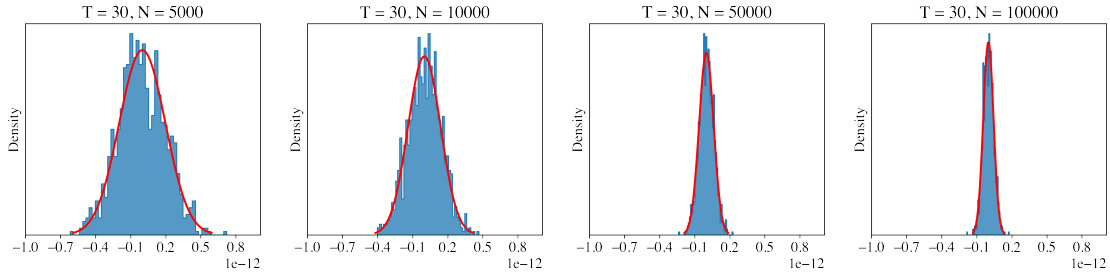
Parameters $\widehat{\mu}, \widehat{\Sigma}$, and $\widehat{K}$ were estimated using training samples generated under the same model. Specifically, N i.i.d. samples $Y_1^{(\mu)}$ were used to estimate $\widehat{\mu}$, N i.i.d. samples of $(Y_1^{(\Sigma)}, Y_2^{(\Sigma)})$ to estimate $\widehat{\Sigma}$, and N i.i.d. samples of $(Y_1^{(K)}, Y_2^{(K)}, Y_3^{(K)})$ to estimate $\widehat{K}$. We chose training sets of size $N = 5000, 10000, 50000, 100000$ and for



(a) Likelihood estimation error from first-order first moment estimation.



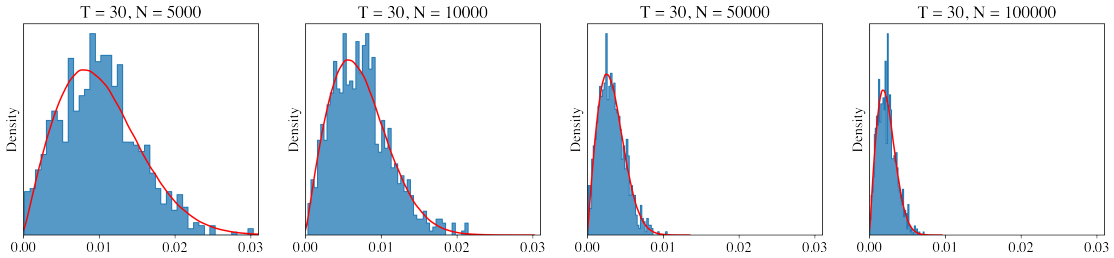
(b) Likelihood estimation error from first-order second moment estimation.



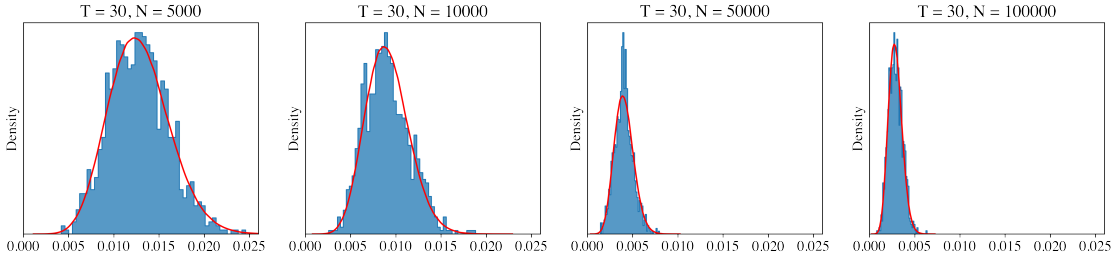
(c) Likelihood estimation error from first-order third moment estimation.

Figure 2.5: Histogram of the first-order error from first, second and third moment estimation error (i.e. $(v+\tilde{v})^\top \widehat{\Delta\mu}$, $\sum_{t=0}^{T} b_t^\top \widehat{\Delta\Sigma} \tilde{b}_t$, and $\sum_{t=1}^{T} a_t^\top \widehat{\Delta K}(y_t)\tilde{a}_t$) under different training size N with length $T = 30$ vs. theoretical pdf calculated based on Theorem 2.2. Histograms are empirical pdf of third moment estimation error, and the red line is the theoretical Normal distribution. Each subfigure is associated with a different N. We could see as N goes larger, the distribution becomes closer to the theoretical normal distribution.

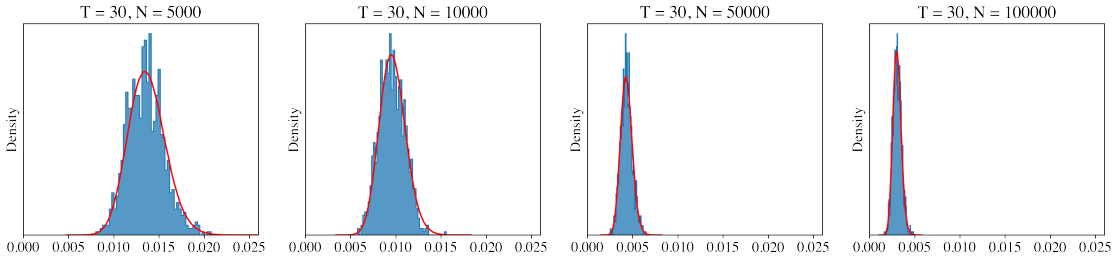
each N we replicated the experiment 1000 times. For each replication, we estimated $\hat{Pr}(x_{1:T})^{(N,r)}$ where r indexes the replication. For each N, we construct the histogram v.s. theoretical pdf for the estimation error, i.e. $\{\hat{Pr}(x_{1:T})^{(N,r)}\}_r$, which by Theorem 2.2 should converge to a normal distribution as N grows larger. In Figure 2.4 we



(a) Frobenius norm of first moment estimation.



(b) Frobenius norm of second moment estimation.



(c) Frobenius norm of third moment estimation error

Figure 2.6: Histogram of the Frobenius norm of the first, second and third moment estimation error (i.e. μ, Σ and K) under different training size N with length $T = 30$ vs. theoretical pdf calculated based on Theorem 2.2. Histograms are empirical pdf of third moment estimation error, and the red line is the theoretical Chi-squared distribution. Each subfigure is associated with a different N. We could see as N goes larger, the distribution becomes closer to the theoretical normal distribution.

indeed see the desired effect. As N grows larger we see that the distribution of the estimated likelihood converges to the normal distribution, and with a shorter length T, the error converges faster. We also separately analyzed the asymptotic behaviour of the first-order estimation error from first moment error, $(v+\tilde{v})^\top \widehat{\Delta\mu}$, second moment error, $\sum_{t=0}^{T} b_t^\top \widehat{\Delta\Sigma} \tilde{b}_t$, and third moment error, $\sum_{t=1}^{T} a_t^\top \widehat{\Delta K}(y_t)\tilde{a}_t$ (as shown in Figure

2.5), and we found that the third moment estimation error dominates the error terms and has the largest contribution as shown in Table 2.1. We further analyzed the asymptotic distribution of Frobenius norm of the first, second and third moment estimation error (i.e. μ, Σ and K). Figure 2.6 shows the empirical histogram and its corresponding theoretical pdf.

We note that there are two facets to the error when estimating the likelihood. The first stems from the typical CLT-type error in estimating the parameters of the model (i.e. for smaller N the $\mathcal{O}_p(N^{-1})$ term is not small enough). The second is that any error introduced into the system from estimation error can propagate under forward recursion. To achieve a stable normal distribution given the second issue, N must be much greater than T. In our simulation ($T = 100$), at $N = 10000$ we see reasonable evidence of asymptotic normality, while at $N = 5000$ we do not. When $N = T$ or N is only slightly larger than T, the distribution is heavy-tailed. That is to say, there could be outliers. This also suggests us to add some kinds of regularization.

For simplicity of presentation, we derived Theorem 2.2 under the assumption that the output distribution is discrete. For continuous output, the central limit theorem still holds with a proper kernel function $G(\cdot)$ as mentioned in section 2.1.3.

2.4 Potential Application Scenario

One possible application scenario for Theorem 2.2 is modeling finance time series. For example, based on the industry reports, quantitative researchers could specify a model that might be momentum (Fama and French, 2012) or mean-reverting (McQueen, 1992), which are the two basic trends of the stock market. With the prior knowledge, the parameters in HMM including initial probability, transition matrix and emissions

Error source	Mathematical expression	Theoretical std	Variance explained ratio
1st moment estimation	$(v+\tilde{v})^\top \widehat{\Delta\mu}$	$1.09 \times 10^{-12}/\sqrt{N}$	0.43%
2nd moment estimation	$\sum_{t=0}^{T} b_t^\top \widehat{\Delta\Sigma} \tilde{b}_t$	$8.98 \times 10^{-12}/\sqrt{N}$	29.21%
3rd moment estimation	$\sum_{t=1}^{T} a_t^\top \widehat{\Delta K}(y_t) \tilde{a}_t$	$1.39 \times 10^{-11}/\sqrt{N}$	70.36%

(a) $T = 30$.

Error source	Mathematical expression	Theoretical std	Variance explained ratio
1st moment estimation	$(v+\tilde{v})^\top \widehat{\Delta\mu}$	$2.66 \times 10^{-49}/\sqrt{N}$	0.30%
2nd moment estimation	$\sum_{t=0}^{T} b_t^\top \widehat{\Delta\Sigma} \tilde{b}_t$	$7.75 \times 10^{-48}/\sqrt{N}$	25.37%
3rd moment estimation	$\sum_{t=1}^{T} a_t^\top \widehat{\Delta K}(y_t) \tilde{a}_t$	$1.33 \times 10^{-47}/\sqrt{N}$	74.59%

(b) $T = 100$.

Table 2.1: Theoretical variance for first, second, third moment estimation errors based on simulated data with different T.

could be explicitly written out, and so does the likelihood associated with this pre-specified model. We could calculate the likelihood estimated from observations with the likelihood calculated from the pre-specified model, and compare them statistically (e.g. through Wald-type Z test, etc), and check whether the observations are likely to be generated from the target model. This could help provide some interpretations for finance time series modeling.

Chapter 3

Projected SHMM

3.1 Motivation for Adding Projection

In the Baum-Welch algorithm (Baum, Petrie, et al., 1970), when we make a prediction, we are effectively predicting the belief probabilities, or weights, for each underlying hidden state. Denote the predicted weight vector at time t as $\hat{w}_t$. Then the prediction can be expressed as a weighted combination of cluster means $\hat{y}_t = M\hat{w}_t$ where $||\hat{w}_t||_1 = 1$. The weights are explicitly guaranteed to be non-negative and sum to 1 during forward propagation in the Baum-Welch algorithm. However, SHMM doesn't estimate the weights directly and doesn't have these two constraints, so SHMM can sometimes give predictions which are far away from the polyhedron spanned by the cluster means, which is inconsistent with the physical meaning of HMM's prediction. Further, when N is not sufficiently large, extreme deviations of the estimated likelihood from the true likelihood can occur if the error is propagated over time. Regularization is needed to stabilize the performance of estimation of the likelihood by limiting this propagation of error. In order to solve this problem, we propose the projected SHMM, where projection serves to regularize the predictions to be within a reasonable range that is consistent with the physical meaning of HMM's prediction.

3.2 Projection-onto-Polyhedron and Projection-onto-Simplex SHMM

One way to regularize the predictions is to project it onto the polyhedron whose vertices are cluster means of different states, which is the domain of prediction yielded by the Baum-Welch algorithm. There are two ways to achieve projection for our problem: projection-onto-polyhedron and projection-onto-simplex. Projection-onto-polyhedron is derived directly from the motivation for using projections in SHMM, but suffers from high computational cost. To obtain a better computational performance, we propose projection-onto-simplex as an alternative, which has the same prediction domain as the projection-onto-polyhedron. We recommend using projection-onto-simplex for PSHMM.

3.2.1 Projection-onto-polyhedron SHMM

Projection-onto-polyhedron PSHMM first predicts $\hat{y}_t^{(SHMM)}$ through the standard SHMM and then projects it onto the polyhedron with vertices $\widehat{M}$, which is the estimated cluster means and in practice approximated by GMM on $\{y_t\}_t$. In other words, we find the point on the polyhedron spanned by $\widehat{M}$ that is nearest to the predicted $\hat{y}_t^{(SHMM)}$. We can use any distance to define "nearest point" but in our exposition we use Euclidean distance. Mathematically, we substitute the recursive forecasting in Equation 2.6 with

$$\begin{aligned} \hat{y}_t^{(SHMM)} &= \frac{C(y_{t-1})\hat{y}_{t-1}}{c_\infty^\top C(y_{t-1})\hat{y}_{t-1}}; \\ \hat{y}_t &= \underset{y \in Poly(\widehat{M})}{\arg\min}\ d(y, \hat{y}_t^{(SHMM)}), \end{aligned} \tag{3.1}$$

where $d(\cdot, \cdot)$ is the distance function (such as Euclidean distance), and

$$Poly(\widehat{M}) = \{y = \widehat{M}w | w \; is \; on \; the \; simplex\}$$

is the polyhedron with vertices $\widehat{M}$. This results in a convex optimization problem if the distance is convex, which is true for general distance functions such as Euclidean distance. We can solve this using standard convex optimization methods such as the Newton-Raphson algorithm (S. Boyd, S. P. Boyd, and Vandenberghe, 2004), with variants allowing linear constraints such as the log-barrier methods (Frisch, 1955). To the best of our knowledge, there is no dedicated algorithm for solving projection-onto-polyhedron, and unfortunately finding a fast solution seems to be challenging. The approach we take is to write the loss function of the constrained problem as the loss function with an indicator function, and use the log-barrier method to approximate the linear constraints through log-barrier functions. We then use the Newton-Raphson algorithm to optimize this approximated loss function, iteratively relaxing the approximation and solving it again until convergence. Note that this optimization needs to be done at every time step, which implies a trade-off between the accuracy of the approximation and optimization. Recall that we turn to SHMM because it is faster than the Baum-Welch algorithm, so any modification should not slow down the computation too much, otherwise this mitigates one of its strongest advantages.

3.2.2 Projection-onto-simplex SHMM

To obtain higher efficiency in computation, we propose an alternative projection regularization method: projection-onto-simplex. It leverages an algorithm that allows us to calculate the projection with time complexity $\mathcal{O}(d \log(d))$ (Wang and Carreira-

Perpinán, 2013). To avoid projection onto a polyhedron, we leverage the fact that $\hat{y}_t = \widehat{M}\hat{w}_t$ and optimize over $\hat{w}_t$ which lies on the simplex. Mathematically, the optimization problem becomes

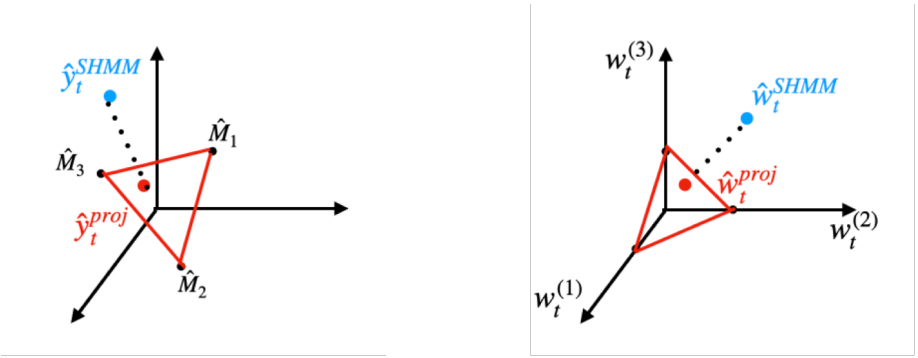
$$\hat{w}_t = \underset{w \in Simplex}{\arg\min} ||w - \widehat{M}^{-1}\hat{y}_t^{(SHMM)}||_2^2. \tag{3.2}$$

This solution is not equivalent to the solution from the projection-onto-polyhedron, because $d(a,b) \neq d(Aa,Ab)$ in general. However, the solution set is the same that the predictions are guaranteed to be constrained to the polyhedron with vertices $\widehat{M}$. The solution of Equation 3.2 can be obtained through a closed-form solution provided in Algorithm 2, which avoids iterations during convex optimization and yields fast estimation. Figure 3.1 gives a graphical demonstration for the projection-onto-polyhedron and projection-onto-simplex methods.

Algorithm 2: Projection-onto-simplex (Wang and Carreira-Perpinán, 2013).

Input : $u = [u_1, u_2, \cdots, u_d]^\top$
Sort u into z: $z_1 \geq z_2 \cdots \geq z_d$;
Find $\rho = max\{1 \leq i \leq d : z_i + \frac{1}{i}(1 - \sum_{j=1}^{i} z_j) > 0\}$;
Define $\lambda = \frac{1}{\rho}(1 - \sum_{j=1}^{\rho} z_j)$;
Solve $u^{(proj)}$, s.t. $u_i^{(proj)} = max(u_i + \lambda, 0)$, $i = 1, \cdots, d$;
Output: $u^{(proj)}$

The full algorithm for PSHMM with projection-onto-simplex is shown in Algorithm 3. In Algorithm 3, Steps 1-3 are identical to the standard SHMM. Steps 4-5 estimate $\widehat{M}$ by GMM, calculate the weight processes $\{w_t\}$, and apply SHMM on the weight process. Step 6 applies projection-onto-simplex on the recursive predicted $\hat{y}_t$. Step 7 projects the data back into the original space.



(a) Projection-onto-polyhedron. (b) Projection-onto-simplex.

Figure 3.1: The left figure shows the projection-onto-polyhedron step, and the right one is the projection-onto-simplex step. In both two methods, we project the predicted values (blue points) into the constrained regions (areas with red boundary), polyhedron (left) or simplex (right).

Algorithm 3: Projection-onto-simplex SHMM.

Input : $\{x_t\}$, where $t = 1, \cdots, T$

Output: $\hat{x}_{T+1}$

Step 1: Compute $\hat{E}[x_{t+1} \otimes x_t] = \frac{1}{T-2} \sum_{i=1}^{T-2} x_{t+1} x_t^\top$;

Step 2: Obtain $\hat{U}$ by extracting the first k left eigenvectors of $\hat{E}[x_{t+1} \otimes x_t]$;

Step 3: Reduce dimensionality $y_t = \hat{U}^\top x_t$;

Step 4: Estimate cluster mean by GMM, and obtain $\hat{M}$, where each column is the mean vector of each cluster. Then the weight vector is $w_t = \hat{M}^{-1} y_t$ for $t = 1, \cdots, T$;

Step 5: Calculate $\hat{\mu} = \frac{1}{T} \sum_{t=1}^{\top} w_t$, $\hat{\Sigma} = \frac{1}{T-1} \sum_{t=1}^{T-1} w_{t+1} w_t^\top$, and $\hat{K} = \frac{1}{T-2} \sum_{t=1}^{T-2} w_{t+2} \otimes w_t \otimes w_{t+1}$. Set $\hat{c}_1 = \hat{\mu}$, $\hat{c}_\infty^\top = c_1^\top \hat{\Sigma}^{-1}$, and $\hat{C}(w_t) = \hat{K}(w_t) \hat{\Sigma}^{-1}$;

Step 6: Recursive prediction with projection-onto-simplex $\hat{w}_t = Proj\left(\frac{\hat{C}(w_{t-1})\hat{w}_{t-1}}{\hat{c}_\infty^\top \hat{C}(w_{t-1})\hat{w}_{t-1}}\right)$ for $t = 2, \cdots, T+1$ where $Proj(a) = \arg\min_{w \in Simplex} ||w - a||_2^2$ can be solved by Algorithm 2, and set $\hat{y}_1 = \hat{c}_1$;

Step 7: $\hat{x}_{T+1} = \hat{U} \hat{y}_{T+1} = \hat{U} \hat{M} \hat{w}_{T+1}$;

3.2.3 Comparison: projection-onto-polyhedron vs. projection-onto-simplex

Projection can pull the prediction within a reasonable range, which serves as a regularization as well as an insanity filtering. There are two ways for the projection. Projection-onto-polyhedron is more intuitive but the optimization is more time-consuming since we have to tune the strength of the constraints and cannot avoid iterations; projection-onto-simplex transfers the problem into optimizing the weight vector, which could leverage a closed-form solution to estimate. In practice, we recommend projection-onto-simplex PSHMM because it can provide an exact solution without tuning the optimization procedure and provide a fast estimation.

3.3 Choice of Hyperparameters and Variants of PSHMM

In this part, we discuss the choice of hyperparameters and then we talk about variations of PSHMM.

3.3.1 The choice of hyperparameter d

d is the dimensionality of the projection space. In theory, d should equal the number of states in the HMM. Our simulations show that when d is chosen to be equal to the underlying true number of states, the estimation and prediction will perform better than at other values of d. However, the number of hidden states is usually unknown in practice, so we can either choose d using prior knowledge or tune it if we do not

have a strong prior belief.

3.3.2 Calculation of U matrix under extremely high-dimensional data: unigram or bigram randomized SVD

The projection matrix U is constructed by the first d left singular vectors from the singular value decomposition (SVD) (Eckart and Young, 1936) of the bigram covariance matrix $\widehat{\Sigma} = \widehat{\mathrm{E}}[X_2 \otimes X_1]$. This encodes the transition information and will eliminate the in-cluster covariance structure. However, this is not the only acceptable projection. For instance, we could also estimate U through an SVD of the unigram covariance matrix $\widehat{\mathrm{E}}[X_1 \otimes X_1]$. This result will encode covariance structure along with the cluster mean information. In most cases we suggest using the bigram matrix.

A point worth mentioning is that for extremely high-dimensional cases, we can leverage a fast approximation algorithm for computing U based on randomized SVD (Halko, Martinsson, and Tropp, 2011). When computing the SVD of the bigram matrix, we need to avoid computing the covariance matrix $\hat{E}[x_{t+1} \otimes x_t]$. The standard algorithm for the SVD requires time complexity $\mathcal{O}(Tp^2 + p^3)$, where T and p are the sample size and dimensionality of the dataset. For the high-dimensional cases where $p \gg d$, the randomized SVD has time complexity $\mathcal{O}(pT\log(d) + (p+T)d^2)$. In this case, the bottleneck is the computation of $\hat{E}[x_{t+1} \otimes x_t]$, whose time complexity is $\mathcal{O}(Tp^2)$. The trick is as follows. Note that $\hat{E}[x_{t+1} \otimes x_t] = \frac{1}{T-2}\sum_{i=1}^{T-2} x_{t+1}x_t^\top = \frac{1}{T-2}X_2^\top X_1$, where $X_2 = [x_2, \cdots, x_T]^\top$ and $X_1 = [x_1, \cdots, x_{T-1}]^\top$. We can take the randomized SVD of X_1 and X_2 separately to obtain two rank-$\tilde{d}$ decompositions with $d \leq \tilde{d} \ll p$: $X_1 \approx U_1\Sigma_1 V_1^\top$, $X_2 \approx U_2\Sigma_2 V_2^\top$. Then $X_2^\top X_1 \approx V_2(\Sigma_2 U_2^\top U_1 \Sigma_1)V_1^\top$. The matrix $(\Sigma_2 U_2^\top U_1 \Sigma_1)$ is of dimension $\tilde{d} \times \tilde{d}$, and computing it is much faster than com-

puting $\hat{E}[x_{t+1} \otimes x_t]$. We could perform SVD on this matrix to get $\Sigma_2 U_2^\top U_1 \Sigma_1 = \tilde{U}\tilde{\Sigma}\tilde{V}^\top$. So $\hat{E}[x_{t+1} \otimes x_t] \approx (V_2\tilde{U})(\frac{1}{T-2}\tilde{\Sigma})(V_1\tilde{V})^\top$. Note that $V_2\tilde{U}$ and $V_1\tilde{V}$ are orthonormal matrices and $\frac{1}{T-2}\tilde{\Sigma}$ is a diagonal matrix, so this is the rank-$\tilde{d}$ SVD of $\hat{E}[x_{t+1} \otimes x_t]$. The first d vectors of $V_2\tilde{U}$ are an approximation of the U matrix we are to compute in Step 1 and 2 in Algorithm 3.

3.3.3 Projecting onto the probability space

Another advantage of PSHMM is that we can project X_t onto the probability space y whose i-th element is the probability that the original data point belongs to the i-th cluster $y_t^{(i)} = \mathrm{P}(h = i|X_t)$. This probability is similar to the emission probability in Gaussian HMM but is now computed by GMM. To do this, we only need to modify Step 4 in Algorithm 3 by substituting $w_t = \widehat{M}^{-1} y_t$ with calculating w_t from the probability of belonging to GMM's different components.

The advantage of this method is interpretation. In this model, w_t has a straightforward probabilistic meaning, i.e. the weight on different components for a given observation. The SHMM prediction is then based on the weight process w_t, while w_t in Algorithm 3 is calculated purely based on the moments. Note that PSHMM with projecting onto the probability space becomes a mixture of moment-based and distribution-based method.

3.4 Simulation

In this section, we test the prediction performance, especially the robustness and the stability, of PSHMM and SHMM under different experimental settings, including

different stickiness of transitions, different signal-noise ratio generated data, misspecified models, and heavy-tailed data.

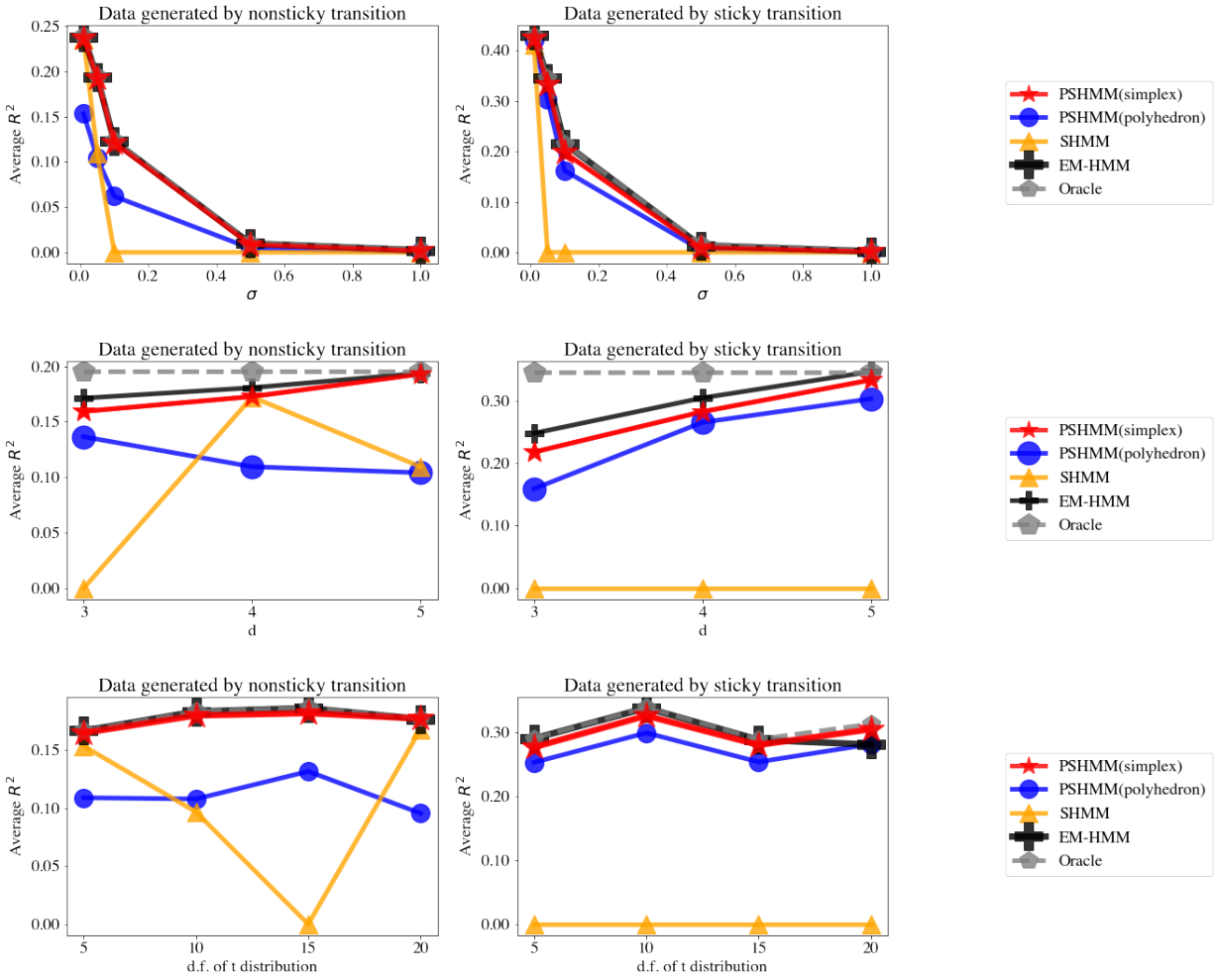


Figure 3.2: These subfigures show the simulation results for experiment settings in Section 3.4. The left column shows the results of the sticky transition, and the right column shows the nonsticky transition. The first row is the results for 5-state GHMM with different σ, the second row is the results for inferring 5-state GHMM of $\sigma = 0.05$ with different d, and the last row is the results for 5-state t-distribution HMM of $\sigma = 0.05$ with different degrees of freedom for t-distribution. In each subfigure, the y-axis is R^2 and different curves are for different methods. For $R^2 < 0$, we plot 0 instead of the negative for plotting purposes. See supplementary for detailed results.

Experiment setting. We generated 100-dimensional data of length 10000 for training data followed by 100 data points for testing under different settings. We used Baum-Welch algorithm with forward propagation and SHMM and PSHMM with recursive prediction for time series forecasting. We repeated each simulation setup 100 times, calculated R^2 for each repeat, and computed an average R^2 over all repeats. The results below show this averaged R^2. We tested 2 variants of the PSHMM: projection-onto-polyhedron and projection-onto-simplex, and compared them with standard HMM and E-M algorithm. The data generated is 5-state continuous output HMM where for each state, we assumed the emission distribution has a one-hot mean vector and diagonal covariance matrix, that the mean vector of the i-th state is $[1\{i = j\}]_{j=1}^{p}$ where $1\{\cdot\}$ is the indicator function. We initialized the first hidden state with equal probabilities. We tested those methods under two types of transition matrix, three different signal-noise ratios and different distributions as below:

- Transition matrix:
  - Sticky transition matrix: diagonal elements are 0.6, off-diagonal elements are $\frac{0.4}{S-1}$, where S is the number of states;
  - Non-sticky transition matrix: diagonal elements are 0.4, off-diagonal elements are $\frac{0.6}{S-1}$.
- Signal-noise ratio: The covariance matrix of each cluster is: $\sigma^2 I_p$, where p is the dimension of the space, $\sigma = 0.01, 0.05, 0.1, 0.5, 1.0$.
- Data generated from:
  - Gaussian distribution: generate according to mean vector and covariance matrix;

- t distribution: generate a standard random vector of *i.i.d.* t_5, t_{10}, t_{15} and t_{20} distribution first, and then multiply by covariance matrix and shifted by mean vector.

For each setting, we also showed the oracle R^2. Strong oracle is to assume we know all parameters and the hidden states before the time stamp to predict. Limited oracle is to assume we know all parameters but don't know the hidden states. Since the limited oracle and the strong oracle are very near, in the plots we only keep the strong oracle as oracle. We will show all detailed simulation results in Appendix.

Simulation results. Figure 3.2 shows the simulation results. We see that adding projections greatly improved R^2 compared with standard SHMM, nearly achieving performance of the oracle in some settings. The top row shows that in both high or low signal-noise ratio case, PSHMM works well. The middle row shows that PSHMM is robust and outperforms SHMM when the model is mis-specified, for example, when the underlying data contains 5 states but we choose to reduce the dimensions to 3 or 4. The last row shows that PSHMM is more robust and has a better R^2 than SHMM with heavy-tailed data generated by t distributions. For $R^2 < 0$ we plot 0 instead of the negative in all these figures for plotting purposes. See supplementary tables for detailed results. Negative R^2 occurs only for SHMM, implying that it is not stable. Overall, while SHMM often performs well, it is not robust, and PSHMM provides a suitable solution.

In all simulation settings, SHMM tends to give poor predictions except in non-sticky and high signal-to-noise ratio settings. PSHMM is robust against noise, mis-specified models and heavy-tailed data. Among the PSHMM variants, projection-onto-simplex outperforms projection-onto-polyhedron, and nearly have the similar performance of

E-M. The reason is that the projection-onto-simplex has a dedicated optimization algorithm that guarantees the exact solution in the projection step. In contrast, projection-onto-polyhedron uses the log-barrier method, which is a general purpose optimization algorithm and does not guarantee the optimality of the solution. Since projection-onto-polyhedron also has a higher computational time, we recommend using projection-onto-simplex. We will show the computational time comparison in the next section.

3.5 Application: Backtesting on High Frequency Crypto-Currency Trading

3.5.1 Data description and experiment setting

To show the performance of our algorithm on real data, we used the crypto currency trading records dataset of Bitcoin, Ethereum, XRP, Cardano and Polygon published by Binance (https://data.binance.vision), one of the largest Bitcoin exchanges in the world. We used the minute-level data, calculated the log return of each minute, then used the log returns as the input for the models. We set aside a test set from 2022-07-01 to 2022-12-31, and for each day in the test set, we used its previous 30-day rolling period to train models (see Figure 3.3), and made consecutive-minute recursive predictions over the testing day without updating model parameters. For prediction, we use the HMM with the E-M algorithm (HMM-EM), SHMM, PSHMM (projection-onto-simplex) and compare their performance. For HMM-EM, SHMM and PSHMM, the hyperparameter d could be tuned on a validation set. We found that $d = 4$ has a better performance and it could be interpreted that there are 4 dominant types of

log returns in general: large loss, small loss, small gain and large gain.

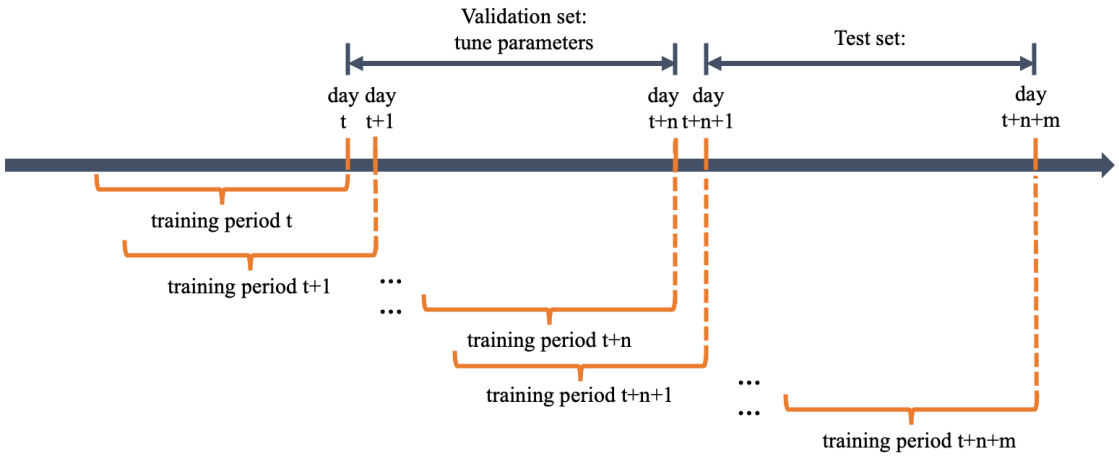


Figure 3.3: Rolling training is used for model validation and testing. For each day of validation and testing sets, the model is trained on the rolling period described in the figure.

Ultimately, we evaluate models based on the performance of a trading strategy. Translating predictions into a simulated trading strategy is straightforward as follows. If we forecast a positive return in the next minute, we buy the currency, and if we forecast a negative return, we short-sell the currency. We buy or short sell a fixed dollar amount of crypto-currency for each of the 5 currencies, hold it for one minute, and then close the position. We repeat this for every minute of the day, and calculate the return of that day as

$$R_m = \frac{1}{5}\sum_{i=1}^{5}\sum_{t} sign(\widehat{Y}_{i,t}^{(m)})Y_{i,t}^{(m)}$$

where $Y_{i,t}^{(m)}$ is the return for minute t of day m for currency i, $\widehat{Y}_{i,t}^{(m)}$ is its prediction, and $sign(a)$ is 1 if a is positive, -1 if a is negative, and 0 if $a = 0$. This backtesting could largely mimic the real-trading procedure, since the target assets are very liquid

and we could ignore the market impact. Market impact is the effect that a market participant has when it buys or sells an asset, including the impact on the capacity, whether the transaction could be done successfully or not.

We evaluate the trading strategy by 3 most widely used metrics: annualized return, Sharpe ratio and maximum drawdown. Over a period of M days, we obtain $R_1, \cdots, R_M$ and calculate the annualized return,

$$Annualized\ return = 365 \times \overline{R},$$

the Sharpe ratio (Sharpe, 1966)

$$Sharpe\ ratio = \frac{\sqrt{365} \times \overline{R}}{\widehat{std}(R)},$$

where $\overline{R}$ and $\widehat{std}(R)$ is the sample mean and standard variance of these daily returns, and the maximum drawdown

$$Maximum\ drawdown = \max_{m_2} \max_{m_1 < m_2} \left[\frac{\sum_{m=m_1}^{m_2} (-R_m)}{1 + \sum_{m=1}^{m_1} R_m} \right].$$

These three metrics are standard mechanisms for evaluating the success of a trading strategy in finance. The annualized return shows the ability of a strategy to generate revenue and is the most straightforward metric. Sharpe ratio is the risk-adjusted return, or the return earned per unit of risk, where the standard deviation of return is viewed as the risk. In general, we can increase both the return and risk by borrowing money or adding leverage, so Sharpe ratio is a better metric than annualized return because it is not affected by the leverage effect. Sharpe ratio is the most commonly used metric in financial literature (see e.g. Falck, Rej, and Thesmar, 2022). Maximum

drawdown is the maximum percentage of decline from the peak. Since the financial data is leptokurtic, the maximum drawdown shows the outlier effect better than the Sharpe ratio which is purely based on the first and second order moments. A smaller maximum drawdown indicates that the method is less risky. It is one of the most widely used metrics for funds' risk control (Grossman and Zhou, 1993; Chekhlov, Uryasev, and Zabarankin, 2004; Chekhlov, Uryasev, and Zabarankin, 2005; Goldberg and Mahmoud, 2017).

3.5.2 Results

Method	Sharpe Ratio	Annualized Return	Maximum drawdown
PSHMM	**2.88**	**1012**%	**49**%
SHMM	1.07	345%	90%
HMM-EM	0.89	197%	53%

Table 3.1: Real-world application results: PSHMM, SHMM, HMM-EM and AR on crypto-currency trading.

From Table 3.1, we see that PSHMM outperforms all other benchmarks with the highest Sharpe ratio and annualized return, and the lowest maximum drawdown. PSHMM outperforms SHMM and SHMM outperforms HMM-EM. SHMM outperforms HMM-EM because the spectral learning doesn't suffer from the local optima problem of the E-M algorithm. PSHMM outperforms SHMM because the projection-onto-simplex provides regularization.

The accumulated daily return is shown in Figure 3.4. PSHMM outperformed other methods. The maximum drawdown of PSHMM is 49%. Considering the high volatility of the crypto currency market during the second half of 2022, this maximum drawdown is acceptable. For computational purposes, the drawdown is allowed to be larger than 100% because we are always using a fixed amount of money to buy or sell,

so effectively we are assuming an infinite pool of cash. Between PSHMM and SHMM, the only difference is from the projection-onto-simplex. We see that the maximum drawdown of PSHMM is only about half that of SHMM, showing that PSHMM takes a relatively small risk, especially given that PSHMM has a much higher return than SHMM. Combining the higher return and lower risk, PSHMM performs substantially better than SHMM.

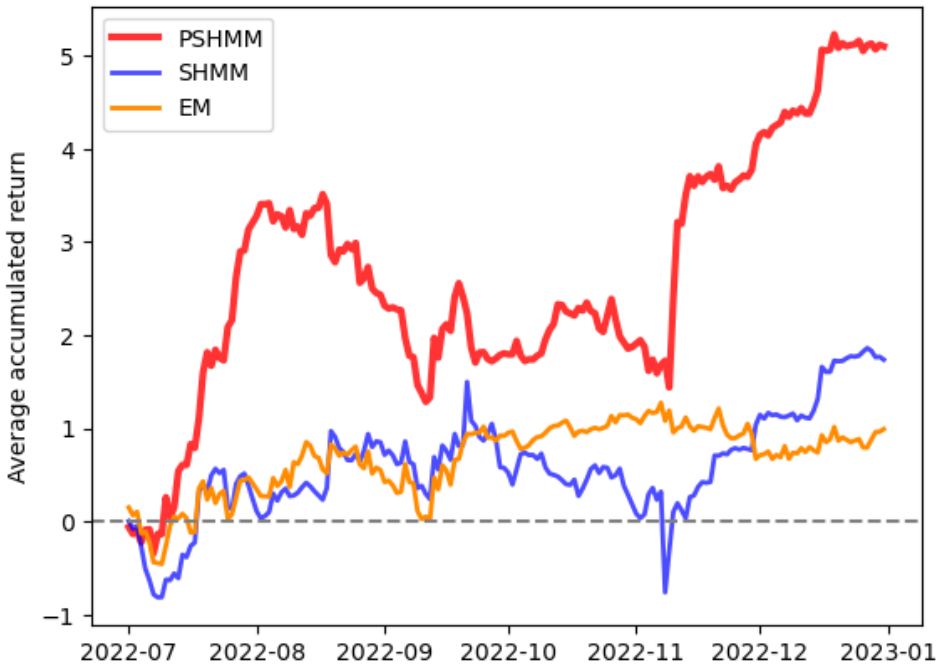


Figure 3.4: The average accumulated return of crypto currencies.

3.6 Discussions

Projection-onto-simplex serves as regularization. The standard SHMM can give poor predictions due to the accumulation and propagation of errors. Projection-onto-simplex pulls the prediction back to a reasonable range. This regularization is our primary methodological innovation, and importantly makes the SHMM well-

suited for practical use.

Bias-variance trade-off of PSHMM In PSHMM, we leverage GMM to provide projection boundaries, which would introduce bias, since the hidden state means estimated by GMM are biased since we ignore time dependency information. In addition, either projection method– 'projection-onto-polyhedron' or 'projection-onto-simplex'– can introduce bias since they are not necessarily an orthogonal projection due to optimization constraints. However, adding such a projection will largely reduce the variance. That is, there is a bias-variance tradeoff.

Chapter 4

Online Learning Variants of SHMM

4.1 Online Learning

As a machine learning algorithm to address the computational complexity issue of Baum-Welch algorithm, it is very natural to apply online learning to SHMM class to accelerate the computational speed. The automatic learning could be categorized into two categories: batch or offline learning and online learning. Batch learning has the following restrictions: (1) the whole data set can be accessed when training (2) we could endure a relatively long computational time and (3) the data generation mechanism doesn't change during the whole process (Fontenla-Romero et al., 2013). However, we found that in the real-world application, there are many application scenarios which cannot meet the above requirements. For example, in quantitative trading, when people want to predict a return of some financial product, it is very likely that the time series is no longer stationary due to the regime switching phenomenon in the market. In addition, in high frequency trading, especially in second-level or minute-level trading, training the statistical model by batch and offline will take more time than online (Lahmiri and Bekiros, 2021) and might yield a delay and impact the strategy and trading speed.

4.1.1 Online learning of SHMM and PSHMM

To adapt the SHMM and PSHMM models as we obtain more data, we first estimate the first, second and third moments based on a warm-up sequnce $\{Y_t\}_{t=1}^{T_{warmup}}$:

$$\begin{aligned}
\widehat{\mu} &\leftarrow \frac{1}{T_{warmup}} \sum_{t=1}^{T_{warmup}} Y_t; \\
\widehat{\Sigma} &\leftarrow \frac{1}{T_{warmup}-1} \sum_{t=1}^{T_{warmup}-1} Y_{t+1} \otimes Y_t; \\
\widehat{K} &\leftarrow \frac{1}{T_{warmup}-2} \sum_{t=1}^{T_{warmup}-2} Y_{t+2} \otimes Y_t \otimes Y_{t+1}; \\
T &\leftarrow T_{warmup}.
\end{aligned}$$

After warm-up, each time we obtain new data point Y_{T+1}, we update our moments as follows:

$$\begin{aligned}
\widehat{\mu} &\leftarrow \frac{T \cdot \widehat{\mu} + Y_{T+1}}{T+1}; \\
\widehat{\Sigma} &\leftarrow \frac{(T-1) \cdot \widehat{\Sigma} + Y_{T+1} \otimes Y_T}{T}; \\
\widehat{K} &\leftarrow \frac{(T-2) \cdot \widehat{K} + Y_{T+1} \otimes Y_{T-1} \otimes Y_T}{T-1}; \\
T &\leftarrow T+1.
\end{aligned} \tag{4.1}$$

The above updating rule works for both SHMM and PSHMM. The pseudo code for online learning PSHMM is shown in Algorithm 4.

For updating GMM for PSHMM To update the first, second and third order moments of w_t, we could just replace Y with w on above formulas. The only consideration is whether to update GMM or not. There are multiple ways of updating GMM.

Algorithm 4: Online learning PSHMM.

Input : $\{x_t\}_{t=1,\cdots,T}$, the warm-up length T_{warmup}
Output: $\{\hat{x}_t\}_{t=T_{warmup}+1}^{T+1}$ yielded sequentially.

Step 1: Compute $\hat{E}[x_{t+1} \otimes x_t]^{(warmup)} = \frac{1}{T_{warmup}-2} \sum_{i=1}^{T_{warmup}-2} x_{t+1} x_t^\top$;

Step 2: Obtain $\hat{U}$ by extracting the first k left eigenvectors of $\hat{E}[x_{t+1} \otimes x_t]^{(warmup)}$;

Step 3: Reduce dimensionality $y_t = \hat{U}^\top x_t$ for $t = 1, \cdots, T_{warmup}$;

Step 4: Estimate cluster mean by GMM by data $\{\hat{y}_t\}_{t=1}^{T_{warmup}}$, and obtain $\hat{M}$, where each column is the mean vector of each cluster. Then the weight vector is $w_t = \hat{M}^{-1} y_t$ for $t = 1, \cdots, T_{warmup}$;

Step 5: Calculate $\hat{\mu}$, $\hat{\Sigma}$, $\hat{K}$, $\hat{c}_1$, $\hat{c}_\infty^\top$ and $\hat{C}(\cdot)$ as described in Step 5 of Algorithm 3;

Step 6: Recursive prediction with projection-onto-simplex $\hat{w}_t$ for $t = 1, \cdots, T_{warmup} + 1$ as described in Step 6 of Algorithm 3. Yield $\hat{x}_{T_{warmup}+1} = \hat{U}\hat{M}\hat{w}_{T_{warmup}+1}$;

Step 7 (online learning and prediction):

for $t \leftarrow T_{warmup} + 1$ **to** T **do**

  $w_t = \hat{M}^{-1}\hat{U}^\top x_t$;

  Update $\hat{\mu}$, $\hat{\Sigma}$ and $\hat{K}$ according to Eq 4.1;

  Predict $\hat{w}_{t+1}$ by $\hat{w}_{t+1} = Proj\left(\frac{\hat{C}(w_t)\hat{w}_t}{\hat{c}_\infty^\top \hat{C}(w_t)\hat{w}_t}\right)$, where $\hat{C}(\cdot)$ and $\hat{c}_\infty^\top$ are based on updated $\hat{\mu}$, $\hat{\Sigma}$ and $\hat{K}$, and $Proj(\cdot)$ is solved by Algorithm 2;

  Yield $\hat{x}_{t+1} = \hat{U}\hat{M}\hat{w}_{t+1}$;

end

We recommend updating it without changing the cluster, for example, we classify a new input into different clusters and update each cluster's mean and covariance. We don't suggest the online learning algorithm of GMM that allows adding or removing clusters. The number of clusters is pre-specified as it should be equal to the number of states in HMM and the dimensionality of space y. The addition of deletion of a cluster in GMM might fundamentally change this requirement. In practice, we found that PSHMM works well without updating GMM.

4.1.2 Online learning of SHMM class with forgetfulness

When dealing with instationary data, it might help if we add a forgetting mechanism on parameter estimation. Particularly, we could add a forgetting mechanism on the moment estimation by using the exponential weighting schema. First we specify a decay factor γ that the information is forgotten by the rate $1-\gamma$ and then the updating rule is

$$\begin{aligned}
\widehat{\mu} &\leftarrow \frac{(1-\gamma)\tilde{T}\widehat{\mu} + Y_{T+1}}{(1-\gamma)\tilde{T}+1}; \\
\widehat{\Sigma} &\leftarrow \frac{(1-\gamma)\tilde{T}\widehat{\Sigma} + Y_{T+1}\otimes Y_T}{(1-\gamma)\tilde{T}+1}; \\
\widehat{K} &\leftarrow \frac{(1-\gamma)\tilde{T}\widehat{K} + Y_{T+1}\otimes Y_{T-1}\otimes Y_T}{(1-\gamma)\tilde{T}+1}; \\
\tilde{T} &\leftarrow \tilde{T}\cdot(1-\gamma)+1.
\end{aligned} \tag{4.2}$$

Here $\tilde{T} = \sum_{i=1}^{T}(1-\gamma)^{i-1}$ serves as an effective sample size. This strategy is equivalent to calculating the exponentially weighted moving average that

$$\begin{aligned}
\widehat{\mu} &\leftarrow \frac{\sum_{t=1}^{T}(1-\gamma)^{T-t}Y_t}{\sum_{t=1}^{T}(1-\gamma)^{T-t}}; \\
\widehat{\Sigma} &\leftarrow \frac{\sum_{t=2}^{T}(1-\gamma)^{T-t}Y_t\otimes Y_{t-1}}{\sum_{t=2}^{T}(1-\gamma)^{T-t}}; \\
\widehat{K} &\leftarrow \frac{\sum_{t=3}^{T}(1-\gamma)^{T-t}Y_t\otimes Y_{t-2}\otimes Y_{t-1}}{\sum_{t=3}^{T}(1-\gamma)^{T-t}}.
\end{aligned} \tag{4.3}$$

The full algorithm is the same as Algorithm 4 expect replacing Equation 4.1 with Equation 4.2 when updating the moment estimation $\widehat{\mu}, \widehat{\Sigma}$ and $\widehat{K}$.

4.2 Simulation

In this section, we show the simulation of online learning for PSHMM. Since online learning is used for further improving the computational speed, so here for PSHMM, we only consider projection-onto-simplex SHMM and don't consider projection-onto-polyhedron SHMM, since we mentioned that projection-onto-polyhedron required an iterative optimization and would largely increase the computational time. Therefore, in this section, we mainly test the predictability and the computational time for online learning PSHMM, and compare it with the online learning SHMM and offline learning SHMM.

4.2.1 Test the prediction performance

In this part, we test the prediction performance of online PSHMM, especially robustness under different signal-noise ratio, mis-specified models and heavy-tailed data.

Experiment setting. The experiment setting for the simulation is the same as the simulation for PSHMM in Chapter 3. See Section 3.4 for more details on simulation configurations and the results below show this averaged R^2. We tested 3 variants of the PSHMM and compared them with the E-M algorithm: offline learning, online learning and online learning with decay factor $\gamma = 5\%$. The online learning variant of PSHMM used 1000 training samples for the initial warm-up, and incorporated the remaining 9000 training samples using online updates. In our simulations, online training for PSHMM differs from offline training for two reasons. First, the estimation of U and $\widehat{M}$ are based only on the warm-up set for online learning (as is the case for the online version of SHMM class), and the entire training set for offline learning.

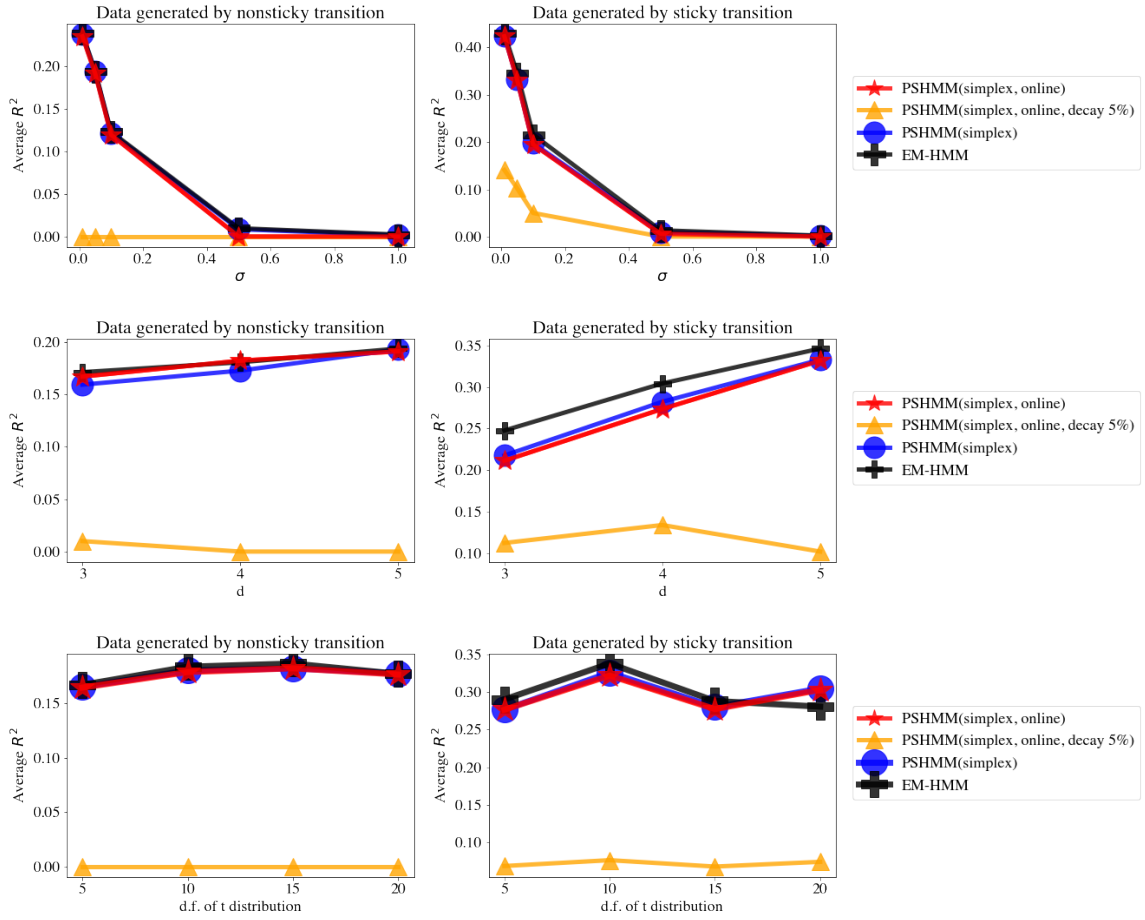


Figure 4.1: These subfigures show the simulation results for online learning experiment settings. The left column shows the results of the sticky transition, and the right column shows the nonsticky transition. The first row is the results for 5-state GHMM with different σ, the second row is the results for inferring 5-state GHMM of $\sigma = 0.05$ with different d, and the last row is the results for 5-state t-distribution HMM of $\sigma = 0.05$ with different degrees of freedom for t-distribution. In each subfigure, the y-axis is R^2 and different curves are for different methods. For $R^2 < 0$, we plot 0 instead of the negative for plotting purposes. See supplementary for detailed results.

Second, during the training period, for PSHMM, the updated moments are based on the recursive predictions of $\hat{w}_t$, whose moment estimators are themselves based on the its previous observations $\{w_s\}_{s=1}^{t-1}$. For offline learning of PSHMM in contrast, all observations are used to calculate the moments.

Simulation results. Figure 4.1 shows the simulation results. We can see that the online learning variant of PSHMM performed as well as PSHMM (the two lines overlapped), which means that online learning didn't sacrifice accuracy. However, if we add a decay factor in the online learning PSHMM, i.e., added forgetfulness, then we could see that the performance was not good. This is because the simulated process is pretty stationary, and added forgetfulness didn't work since there are no new generated patterns, and added forgetfulness means the model would be changed a lot. However, in real-world application, the stationary assumption is very easily violated. Therefore, modeling training with forgetfulness will become very necessary, which we show later.

4.2.2 Test computational time of online learning variants

Experiment setting. We used a similar experimental setting from Section 3.4. We simulated 100-dimensional, 3-state GHMM data with $\sigma = 0.05$ and of length 2000. We use the first half for warm-up and test computational time on the last 1000 time steps. We tested both the E-M algorithm and SHMM. For SHMM, we tested under both online and offline learning regimes. We computed the total running time in seconds. The implementation is done in python with packages Numpy (Harris et al., 2020), Scipy (Virtanen et al., 2020) and scikit-learn (Pedregosa et al., 2011) without multithreading. The whole process is repeated 30 times, and the computational time

is the average of each time.

Simulation results. Table 4.1 shows the computational time. First, online learning substantially reduces the computational cost. For the offline learning methods, projection-onto-simplex SHMM performs similarly with SHMM, and projection-onto-polyhedron is much slower. In fact, the offline version of projection-onto-polyhedron is slow even compared to the Baum-Welch algorithm. However, the online learning variant of projection-onto-polyhedron is much faster than the Baum-Welch algorithm. Taking both the computational time and prediction accuracy into consideration, online and offline projection-onto-simplex SHMM are the best choice among these methods.

Method	Offline/online	Computational time (sec)
EM (Baum-Welch)	-	2134
SHMM	offline	304
SHMM	online	0.5
PSHMM (simplex)	offline	521
PSHMM (simplex)	**online**	**0.7**
PSHMM (polyhedron)	offline	10178
PSHMM (polyhedron)	online	14

Table 4.1: Simulation results for comparing computational time among different methods.

4.2.3 Test the effectiveness of forgetfulness

Experiment setting. Similar to Section 3.4 and Section 4.2.1, we simulated 100-dimensional, 5-state GHMM data with $\sigma = \{0.01, 0.05, 0.1, 0.5, 1.0\}$ and of length $1000 + 1000$ where the first 1000 steps are for training and the last 1000 steps are for testing. The transition matrix is no longer time-constant but differ in the training and testing period as follows:

- training period (diagonal-0.8):

$$\begin{bmatrix} 0.8 & 0.05 & 0.05 & 0.05 & 0.05 \\ 0.05 & 0.8 & 0.05 & 0.05 & 0.05 \\ 0.05 & 0.05 & 0.8 & 0.05 & 0.05 \\ 0.05 & 0.05 & 0.05 & 0.8 & 0.05 \\ 0.05 & 0.05 & 0.05 & 0.05 & 0.8 \end{bmatrix};$$

- testing period (antidiagonal-0.8):

$$\begin{bmatrix} 0.8 & 0.05 & 0.05 & 0.05 & 0.05 \\ 0.05 & 0.8 & 0.05 & 0.05 & 0.05 \\ 0.05 & 0.05 & 0.8 & 0.05 & 0.05 \\ 0.05 & 0.05 & 0.05 & 0.8 & 0.05 \\ 0.05 & 0.05 & 0.05 & 0.05 & 0.8 \end{bmatrix}.$$

We tested different methods on the last 100 time steps with four methods including standard SHMM, projection-onto-simplex SHMM, online learning projection-onto-simplex SHMM, and online learning projection-onto-simplex SHMM with decay factor $\gamma = 0.05$. For online learning methods, similar to Section 4.2.1, we used the first 100 in the training set for warm-up and incorporated the remaining 900 samples by online updates.

Simulation results. Figure 4.2 shows the simulation results. As we can see, when the underlying data generation process is no longer with a stationary scheme, most methods failed including the E-M algorithm, except the online learning projection-onto-simplex SHMM with decay factor = 0.05. This indicated that adding the decay

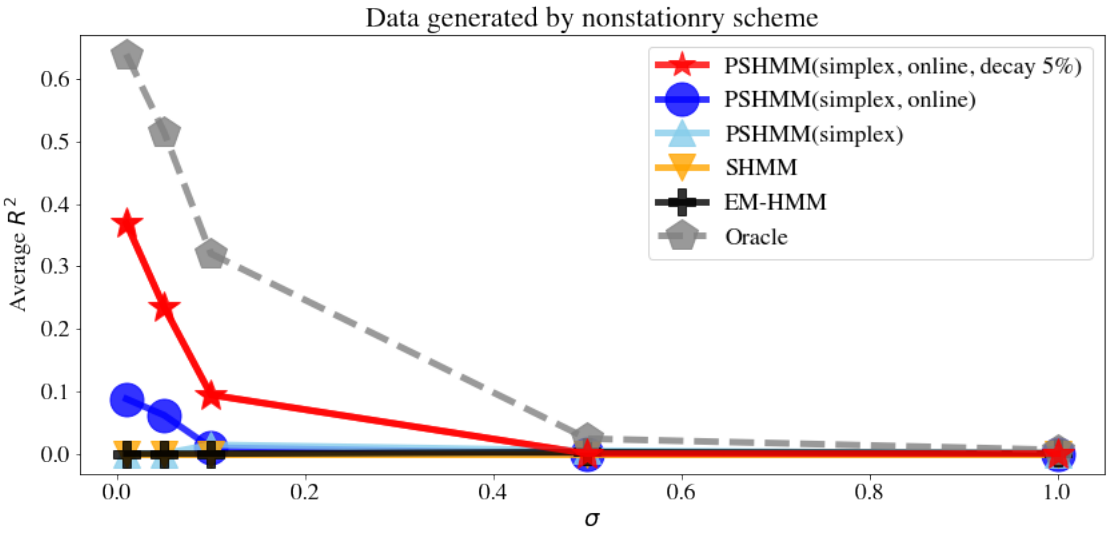


Figure 4.2: The figure shows the simulation results for online learning variants. The results are for 5-state GHMM with different σ and time-varying transitions. The y-axis is R^2 and different curves are for different methods. For $R^2 < 0$, we plot 0 instead of the negative for plotting purposes. See supplementary for detailed results.

factor could help the model adapt to the non-stationary pattern, and will have more robust results. At the same time, we could see that when the signal noise ratio is small, online learning with decay doesn't work well. This is due to the limited effective sample size.

4.3 Application: Backtesting on Commodity Market Daily Trading

4.3.1 Data description & experiment setting

Commodity future contract is a very important investment class in the world. Here we are using the historic daily commodity future contract close price to make fore-

casting. The data is downloaded from Yahoo Finance. We chose 8 most popular assets including crude oil, natural gas, gold, wheat, S&P 500 Index, Dow Jones Index, Nasdaq 100 Index and US treasury bond. We first calculated the log return of the close price for each day, then used it as the input of models. We set a validation period from 09/01/2021 to 12/31/2021, and then set the whole year 2022 as a test set. For each day in the validation and testing set, we used its previous 1-year (252 trading days) rolling period as training. We tuned the hyperparameters d from 3 to 6 over the validation based on Sharpe ratio and applied the optimal one to the test set. In the test set, we convert the prediction into trading strategies and measure its performance by annualized returns, Sharpe ratios and maximum drawdowns as in Section 3.5.1. We test PSHMM and SHMM under 3 settings: offline learning, online learning and online learning with decay $\gamma = 0.05$; we also compare them with HMM-EM.

4.3.2 Results

The results are shown in Table 4.2. From the results, we can see that the PSHMM outperformed the benchmarks such as E-M algorithm and standard SHMM. The offline training PSHMM is better compared to online learning since it could obtain the whole set of training at one time, so the estimation of moments is more accurate at the beginning stage. Since we only use a 1-year rolling training set, the dynamic change is not severe in general. Online learning PSHMM with 5% decay has a decent performance, which shows the adaptability to the dynamic nature of the trading data.

Method	Sharpe Ratio	Annualized Return	Maximum drawdown
PSHMM	**2.563**	**3.498**	**0.257**
online PSHMM	0.600	0.827	0.631
online PSHMM decay 5%	**1.366**	**1.873**	**0.226**
SHMM	-0.146	-0.201	1.152
online SHMM	0.342	0.490	0.946
online SHMM decay 5%	-0.484	-0.664	1.101
HMM-EM	0.187	0.278	0.575

Table 4.2: Real-world application results: SHMM class with online learning on commodity trading.

4.4 Discussions

Online learning can adapt to dynamic patterns and provide faster learning. Here we provide an online learning strategy that allows the estimated moments to adapt over time, which is critical in several applications that can exhibit nonstationarity. Our online learning framework can be applied to both the standard SHMM and PSHMM. Importantly, online learning substantially reduces the computational costs compared to re-training the entire model prior to each new prediction.

Trade-off for using online learning with decay. Whether to use a nonzero decay factor γ for online learning depends on several factors. The biggest factor is whether the pattern is dynamic or not. If the pattern is dynamic, we should also consider the signal-noise-ratio. When we are using online learning with decay, the effective sample size, that the summation of all weights are limited even if we have infinite samples. If we have infinite samples, the summation of weights is $\sum_0^\infty (1-\gamma)^t = \frac{1}{\gamma}$. That is to say, if we choose the decay factor $\gamma = 0.05$, then the effective sample size is about 20; if we choose the decay factor $\gamma = 0.01$, then the effective sample size is about 100. Unlike the non-decay learning whose sample size goes to infinite when we have infinite observations, the effective sample size is bounded if the

decay is nonzero. So if the pattern is only slightly dynamic, we could still choose to use the non-decay online learning.

Chapter 5

Latent Control Hidden Markov Models

5.1 Motivation

Previous chapters introduce the estimation of standard HMM evolving endogenously with a strong stationary assumption. However, in many cases, the evolution of the hidden states does not follow a time-homogeneous Markov chain, meaning that the state transition probabilities are not constant over time. In this study, we focus on HMM whose transition is impacted by the cross predictors outside the system and extend it to a Bayesian model by adding priors. However, this model is not designed to deal with sparse models that some features are not incorporated in the models.

To deal with sparsity, we propose a new method called latent control HMM, with latent binary variables controlling the state transitions by automatically selecting the cross predictors in and out of the model. These binary latent variables further form spike-and-slab priors (Ishwaran and Rao, 2005a) and could indicate how the corresponding feature impacts the transition with providing a posterior probability. The model inference could be done by MCMC sampling. For simple models like linear regression with spike-and-slab prior, inferring the latent control variable could be done with sampling from the conjugate class. However, for some more complicated models

such as the proposed latent control HMM, the conjugate class is hard to find, a more common strategy is to do the sampling with Metropolis-Hasting sampler (Chib and Greenberg, 1995). MCMC could help us predict better under sparse cases or under the cases where the exogenous features only weakly impact the transition matrix. When prediction, we could sample from the posterior by MCMC and then take the average. This could be viewed as a Bayesian model averaging. Besides prediction, these MCMC samples can provide the feature importance by providing the posterior probability for each feature of being selected in the model.

To summarize, we propose a latent control HMM with a latent binary variable, which could provide insights on feature importance on transition between states. We propose to use MCMC to infer its parameters and make predictions. We tested this model on both simulated data and application data with benchmark methods, and we found that the latent control model has a better performance than other models.

5.2 Literature Review

5.2.1 Bayesian HMM

Bayesian HMM is an hierarchical model that every parameter including initial probability π_0, transition matrix T and emission E in standard HMM follows some priors

as follows:

$$
\begin{aligned}
\pi_0 &= [\pi_0^1, \pi_0^2, \cdots, \pi_0^S] \sim Dir(1, \cdots, 1), \\
h_1|\pi_0 &\sim Multinomial(1, \pi_0), \quad h_1 = 1, \cdots, S, \\
T &= \begin{bmatrix} T_{1,1} & T_{1,2} & \cdots & T_{1,S} \\ T_{2,1} & T_{2,2} & \cdots & T_{2,S} \\ \cdots & \cdots & \cdots & \cdots \\ T_{S,1} & T_{S,2} & \cdots & T_{S,S} \end{bmatrix} \text{ where } T_{(i,\cdot)} \sim Dir(1, \cdots, 1), \\
(\mu_i, \Sigma_i) &\sim NIG(\mu_0, N_0, c, d), \quad i = 1, \cdots, S, \\
Y_{it} &\sim N(\mu_{h_{it}}, \Sigma_{h_{it}}),
\end{aligned} \tag{5.1}
$$

where $T_{i,j}$ indicates the transition probability from hidden state i to j, $T_{(i,\cdot)}$ refers to the transition probability vector of transiting from hidden state i to all next possible hidden states, and NIG is the Normal-Inverse-Gamma distribution. For simplicity, we denote the set of Gaussian emission distribution parameters as $E = \{\mu_i, \Sigma_i\}_{i=1:S}$. Bayesian HMM by Markov chain Monte Carlo (MCMC) is a very straightforward by Gibbs sampling (see e.g. Rydén, 2008).

5.2.2 HMM with cross predictors: control HMM

Control HMM was proposed multiple times in the 1990s. For example, Bengio and Frasconi (1994) proposed the input-output HMM and in this framework, researchers introduced how to add the cross predictors to the standard HMM. Figure 5.1 shows the model structure. Compared with the standard HMM, the transition matrix in the control HMM can be impacted by the exogenous features, and the transition is no longer a constant due to incorporating time-dependent variables. Typically, the

transition matrix is typically in the multinomial logistic form as follows:

$$
\begin{aligned}
T_{it}(r,s) &= P(H_{it}=s|H_{(i,t-1)}=r) = \frac{exp(X_{it}\beta_{rs})}{1+\sum_{k=2}^{S} exp(X_{it}\beta_{rs})}, \\
T_{it} &= \begin{pmatrix} B_1 & 0 & \dots & 0 \\ 0 & B_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & B_S \end{pmatrix} \begin{pmatrix} 1 & exp(X_{it}\beta_{12}) & \dots & exp(X_{it}\beta_{1S}) \\ 1 & exp(X_{it}\beta_{22}) & \dots & exp(X_{it}\beta_{2S}) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & exp(X_{it}\beta_{S2}) & \dots & exp(X_{it}\beta_{SS}) \end{pmatrix} \qquad (5.2)
\end{aligned}
$$

where $B_b = 1 + \sum_{k=2}^{S} exp(X_{it}\beta_{rs})$ is the normalization term. This model can be inferred by the E-M algorithm for MLE and Algorithm 5 shows the full algorithm of using E-M algorithm to infer the control HMM.

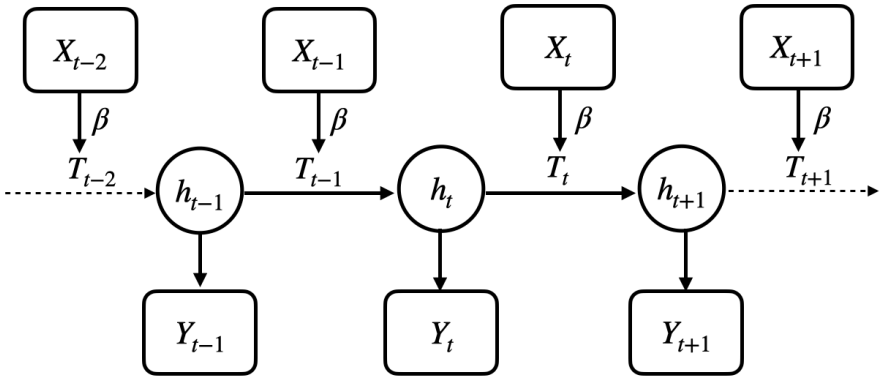


Figure 5.1: Control HMM with MLE by (Bengio and Frasconi, 1994)

5.3 Model Assumption

In this section, we described the proposed latent control HMM. Assume Y_t follows a HMM whose transition is controlled by exogenous features X_t. Assume X_t is p-dimensional where p is the number of cross predictors, Y_t is q-dimensional where q is

Algorithm 5: E-M algorithm for HMM with cross predictors

Data: $\{Y_t\}_{t=1}^T$, $\{X_t\}_{t=1}^T$
Result: $\widehat{\Theta} = \{\widehat{\pi}_0, \widehat{\beta}, \widehat{\mathcal{E}}\}$
Initialize $\widehat{\Theta}$;
while *not converged* **do**
  E-step: update $\{\gamma_t(j)\}_{t=1:T}^{j=1:S}$ and $\{\xi_t(i,j)\}_{t=1:T}^{i,j=1:S}$ conditioned on $\widehat{\Theta}$ **for** $t \leftarrow 1$ **to** T **do**
    $$\alpha_t(j) = \begin{cases} \pi_0(j) b_j(Y_1), & \text{if } t = 1 \\ \sum_{i=1}^S \alpha_{t-1}(i) \boldsymbol{T}_{ij}(X_{t-1}; \boldsymbol{\beta}) b_j(Y_t), & \text{otherwise} \end{cases} \quad \forall j = 1, \cdots, S;$$
  end
  for $t \leftarrow T$ **to** 1 **do**
    $$\beta_t(j) = \begin{cases} 1, & \text{if } t = T \\ \sum_{j=1}^S \boldsymbol{T}_{ij}(X_t; \boldsymbol{\beta}) \beta_{t+1}(j) b_j(Y_{t+1}), & \text{otherwise} \end{cases} \quad \forall j = 1, \cdots, S;$$
  end
  $\mathrm{P}(h_t = j) = \gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{\sum_{i=1}^S \alpha_t(i)\beta_t(i)}$ $\forall t, j$;
  $\mathrm{P}(h_t = i, h_{t+1} = j) = \xi_t(i,j) = \frac{\alpha_t(i) T_{ij}(X_t;\beta) b_j(X_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^S \alpha_t(i)\beta_t(i)}$ $\forall t, i, j$;
  M-step: update $\widehat{\Theta}$ conditioned on $\{\gamma_t(j)\}$ and $\{\xi_t(i,j)\}$
  for $i \leftarrow 1$ **to** S **do**
    $X^{(logit)}, y^{(logit)}, w^{(logit)} \leftarrow$ *empty array*;
    for $t \leftarrow 1$ **to** T **do**
      for $j \leftarrow 1$ **to** S **do**
        append$(X^{(logit)}, X_t)$;
        append$(y^{(logit)}, j)$;
        append$(w^{(logit)}, \xi_t(i,j))$;
      end
    end
    Fit multinomial-logistic regression with $X^{(logit)}$ as features, $y^{(logit)}$ as responses, and $w^{(logit)}$ as sample weights;
    $\widehat{\boldsymbol{\beta}}_{i\cdot}$ is its regression coefficient;
  end
  $\widehat{\pi}_0(j) = \gamma_1(j)$ $\forall j = 1, \cdots, S$;
  Estimate $\widehat{\mathcal{E}}_j$ based on samples $\{Y_t\}_{t=1}^T$ with weights $\{\gamma_t(j)\}_{t=1}^T$
end

the number of response, (X_t, Y_t) are both observed at time t.

Assume there are S hidden states, and define the part related to π_0 as below:

$$\begin{aligned} \pi_0 &= [\pi_0^1, \pi_0^2, \cdots, \pi_0^S] \sim Dir(1, \cdots, 1), \\ h_1|\pi_0 &\sim Multinomial(1, \pi_0), \quad h_1 = 1, \cdots, S \end{aligned} \tag{5.3}$$

where π_0 is the initial probability of hidden states in Equation 5.3, and hidden states h_t is a categorical variable with levels $1, \cdots, S$. Here we slightly abused the definition of Multinomial distribution that we want the notation of the model to be simple. h_t could be of the format of either an one-hot dummy variable or an integer $1, \cdots, S$.

Second, the latent control variable and the coefficient of cross predictors have a continuous bimodal spike-and-slab structure as follows:

$$\begin{aligned} w &\sim Beta(a, b), \\ Z_{rs}^m &\sim Bern(w), \quad m = 1, \cdots, p; r = 1, \cdots, S; s = 2, \cdots, S \\ \beta_{rs} &= [\beta_{rs}^1, \ldots, \beta_{rs}^p], \\ \beta_{rs}^m|(Z_{rs}^m = 0) &\sim Normal(0, \sigma_{spike}^2), \\ \beta_{rs}^m|(Z_{rs}^m = 1) &\sim Normal(0, \sigma_{slab}^2), \end{aligned} \tag{5.4}$$

where $Z = \{Z_{rs}^m\}_{mrs}$ is the latent variable, which controls $\beta = \{\beta_{rs}^m\}_{mrs}$, and Z and β are of the same dimensionality $p \times S \times S$, where m is the m-th predictor among p cross predictors, Z_{rs}^m controls the scale of β_{rs}^m by spike-and-slab prior, and β_{rs}^m controls the impact of the m-th feature on the transition from hidden state r to s. By default, in Equation 5.4 we set priors $a = b = 1$, $\sigma_{spike}^2 = \frac{1}{1000}$ and $\sigma_{slab}^2 = 1$.

Then, define the transition matrix and hidden states:

$$
\begin{aligned}
T_{it}(r,s) &= P(h_{it}=s|h_{(i,t-1)}=r) = \frac{exp(X_{it}\beta_{rs})}{\sum_{k=1}^{S} exp(X_{it}\beta_{rk})}, \\
T_{it} &= \begin{pmatrix} B_{it,1} & 0 & \dots & 0 \\ 0 & B_{it,2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & B_{it,S} \end{pmatrix} \begin{pmatrix} exp(X_{it}\beta_{11}) & exp(X_{it}\beta_{12}) & \dots & exp(X_{it}\beta_{1S}) \\ exp(X_{it}\beta_{21}) & exp(X_{it}\beta_{22}) & \dots & exp(X_{it}\beta_{2S}) \\ \vdots & \vdots & \ddots & \vdots \\ exp(X_{it}\beta_{S1}) & exp(X_{it}\beta_{S2}) & \dots & exp(X_{it}\beta_{SS}) \end{pmatrix}, \\
h_{i(t+1)} &\sim Multinomial(1, T_{it}(h_{it},\cdot)), \qquad (5.5)
\end{aligned}
$$

where T_{it} is the transition matrix for each observation i at each time step t, and $T_{it}(r,s)$ is the elements representing the transition probability from hidden state r to s in the transition matrix T_{it}. The transition matrix is in the multinomial-logistic format. In Equation 5.5, $B_{it,s}$ is the normalization term and $B_{it,s} = \frac{1}{\sum_{k=1}^{S} exp(X_{it}\beta_{rk})}$. $T_{it}(h_{it},\cdot)$ represents the probability vector of hidden states transiting from h_{it} to each hidden state.

Finally define the part for emission and observations:

$$
\begin{aligned}
E &= \{\mu_i, \Sigma_i\}_{i=1:S} \text{ where } (\mu_i,\Sigma_i) \sim NIG(\mu_0, N_0, c, d), \quad i = 1, \cdots, S, \\
Y_{it} &\sim N(\mu_{h_{it}}, \Sigma_{h_{it}}), \qquad (5.6)
\end{aligned}
$$

where E is the set of parameters representing the Gaussian emission distributions, which follow a Normal - Inverse Gamma prior distribution with hyperparameters $(\mu_0, N_0, c, d,)$, where μ_0 is the prior mean for μ_i, N_0 is the prior strength, and c and d are the parameters for $\Sigma_{h_{it}}$. By default we set $\mu_0 = 0$, $N_0 = 10$, $c = d = 1$.

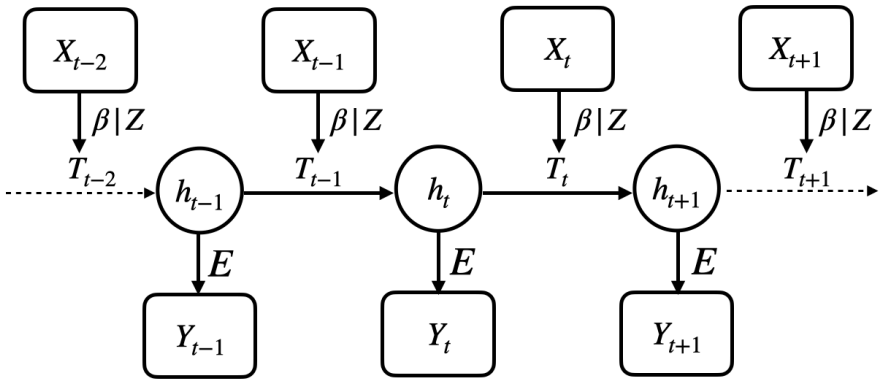


Figure 5.2: Latent control HMM model structure. $\beta = \{\beta_{rs}^m\}$, $Z = \{Z_{rs}^m\}$, $m = 1, \cdots, p$, $r, s = 1, \cdots, S$.β_{rs}^m and Z_{rs}^m is 1-1 mapping, and β and Z are not time-dependent and will not change over time.

5.4 Model Inference

5.4.1 Model estimation by MCMC

To infer the parameters in the model, we used Gibbs MCMC sampling with stochastic search variable selection (Ishwaran and Rao, 2005b) to infer w and $\{Z_{rs}^m\}_{mrs}$ and Metropolis-Hasting sampler for inferring $\{\beta_{rs}^m\}_{mrs}$. We divided the inference into six main steps as following:

- Initialization: set initial value for parameters, including π_0, E and β_{rs}^m by MLE;
- Step 1: Sample $h_1, h_2, \cdots, h_T$

  Sample h_t based on the posterior, and get the probability $p(h_t = s)$ for each possible hidden state. Sample the hidden states based on Multinomial distribution and backward recursion forward sampling.
- Step 2: Sample w

Sample w from posterior distribution $p(w|Z_{rs}^m)$, where $w|Z_{rs}^m \sim Beta(a+\#\{Z_{rs}^m = 0\}, b+\#\{Z_{rs}^m \neq 0\})$.

- Step 3: Sample Z_{rs}^m

  Sample Z from continuous bimodal distribution as follows:

$$\begin{cases} P(Z_{rs}^m = 0) \propto (1-w)(\frac{\sigma_{spike}^2}{\sigma_{slab}^2})^{-\frac{1}{2}} exp(-\frac{(\beta_{rs}^m)^2}{2\sigma_{spike}^2}) \\ P(Z_{rs}^m = 1) \propto w exp(-\frac{(\beta_{rs}^m)^2}{2\sigma_{slab}^2}) \end{cases},$$

- Step 4: Sample β_{rs}^m by Metropolis-Hasting

  Propose a new β_{rs}^m by Gaussian random walk, then calculate the posterior probability. If the posterior becomes larger, then accept the proposed β_{rs}^m; otherwise, accept the proposal with an acceptance rate.

- Step 5: Sample E:

  Sample $\mu_i, \Sigma_i, i = 1, \cdots, S$ based on Normal-Inverse Gamma conjugate posterior distribution.

- Step 6: Sample π_0:

  Sample π_0 by the Dirichlet-Dirichlet conjugate. See more details.

- Repeat Step 1 to Step 6 until all parameters involved in this model for enough sampling times (e.g. 1000).

The above sampling procedure is for a single observed series, and the generalization to multiple observed series is straightforward. Here the biggest difference between this method and traditional Bayesian HMM's MCMC sampling is that we don't have the posterior distribution for coefficients β, so we proposed to use Metropolis-Hasting sampler. The mathematical details of the above procedure is as follows.

Sampling $h_1, \cdots, h_T$ by backward recursion forward sampling For each observation i, sample hidden states $h_1, \cdots, h_T$ as based on the following probability.

$$\begin{aligned} P(h_1 = r|\cdots) &\propto \pi_0^r \phi(y_{i1}; \mu_r, \sigma_r) p(y_{i2:iT}|h_1 = r), \\ P(h_t = r|h_{t-1} = s) &\propto T_{i(t-1)}(s, r)\phi(y_{it}; \mu_r, \sigma_r) p(y_{i(t+1):iT}|h_t = r), \end{aligned} \tag{5.7}$$

where $\phi(\cdot)$ is the Gaussian pdf, and $p(y_{i(t+1):iT}|h_t = r)$ is the backward propagation probability. That is to say, we sample h_1 based on $P(h_1 = r|\cdots)$ in Eq. 5.7, and then conditioned on the sampled h_1, sample h_2. Then conditioned on sampled h_2, sample h_3. Recursively sample until h_T.

Metropolis-Hasting algorithm for sampling exogenous predictor coefficients Suppose we have a previous sample β_{rs}^m and a proposed $\widetilde{\beta_{rs}^m}$. We calculate the posterior, if the posterior of the new proposal is larger than the old one, then accept and update the previous β_{rs}^m by it. Otherwise, accept the proposed parameter with the acceptance rate in Eq. 5.8.

$$Acceptance\ Rate = min\{1, \frac{\pi(\widetilde{\beta_{rs}^m}) L(\widetilde{\beta_{rs}^m}, \cdots)}{\pi(\beta_{rs}^m) L(\beta_{rs}^m, \cdots)}\} = min\{1, H\}, \tag{5.8}$$

where $H = \frac{\pi(\widetilde{\beta_{rs}^m}) L(\widetilde{\beta_{rs}^m}, \cdots)}{\pi(\beta_{rs}^m) L(\beta_{rs}^m, \cdots)}$ is the Hasting rate. Then we sample a uniformly distributed variable $u \sim Unif[0, 1]$ and update β_{rs}^m to $\widetilde{\beta_{rs}^m}$ if we have $u \leq min\{1, H\}$, else we keep β_{rs}^m. The proposal $\widetilde{\beta_{rs}^m}$ can be sampled from a Gaussian random walk around the previous β_{rs}^m.

Sample the initial probability by Dirichlet distribution Sample $\pi_0^{updated}$ from the posterior as following:

$$\pi_0^{updated} \sim Dir(1 + \#(h_1 = 1), \cdots, 1 + \#(h_1 = S)), \tag{5.9}$$

where h_1 is the hidden states at time step $t = 1$.

5.4.2 Prediction & Bayesian credible interval

Given the sampled parameter set $\Theta = (\pi_0, E, \beta, Z)$, we can use the standard forward propagation to make predictions based on the latent-control HMM model. Figure 5.3 shows the prediction steps. Assume we use time steps $1, \cdots, t-1$ as training set for MCMC parameter sampling, and use time steps $t, \cdots, T$ as test set. For each time step t, conditioned on all information before t, at time t, the hidden state $\hat{h}_t|t-1 = h_{t-1}T_{t-1}$. The predicting at time step t is $\hat{Y}_t = (\hat{h}_t|t-1) \cdot \vec{\mu}$. When predict the hidden state h_{t+1}, we need to update the observation of Y_t and also update h_t based on forward-propagation and repeat the above steps. These steps are the same as prediction by frequentist HMM as all parameters are fixed. Note that we have M different sampled parameters by MCMC $\{\Theta_n\}_{n=1}^M$, following the posterior distribution. For each of these parameter sets, we will get a prediction $\widehat{Y}_t|\Theta_m$ for each time step t, and their arithmetic average of M predictions are the final prediction $\widehat{Y}_t^{(MCMC)} = \frac{1}{M}\sum_{n=1}^M(\widehat{Y}_t|\Theta_n)$, which is the posterior mean prediction. Based on the MCMC samples, we can construct 95% Bayesian credible interval (Eberly and Casella, 2003) by finding the empirical 2.5% and 97.5% quantiles of $\{(\widehat{Y}_t|\Theta_n)\}_{n=1}^M$.

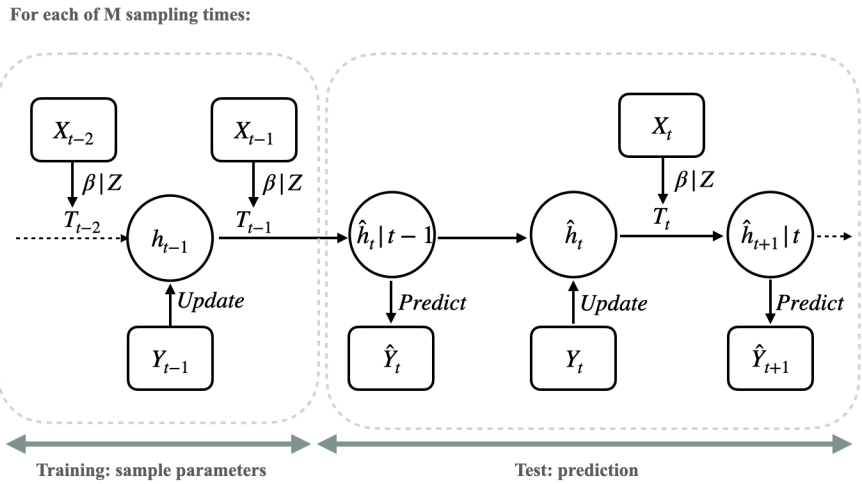


Figure 5.3: Prediction by latent control HMM

5.4.3 Feature importance by posterior mean

The most interesting part of latent control HMM is that it can provide a way to evaluate feature importance by the posterior mean. This feature importance mechanism we proposed is unique to the best of our knowledge. The motivation is that if $Z_{rs}^m = 1$, then β_{rs}^m is in the slab mode, otherwise in the spike mode. We can treat the β_{rs}^m in the slab mode to be in the model and treat the β_{rs}^m in the spike mode to be out of the model. So we find the posterior probability of $Z_{rs}^m = 1$ from MCMC samples as the posterior probability that β_{rs}^m is in the model.

Mathematically, after the MCMC sampling, for the m-th feature on transition from state r to s, we have $\{(Z_{rs}^m)_n\}_{n=1}^M$ that is the MCMC samples for whether this feature could impact this transition. We calculate the posterior mean that $Pr_D(Z_{rs}^m) = \overline{Z_{rs}^m} = \frac{1}{M}\sum_{n=1}^M m(Z_{rs}^m)_n$ as the feature importance score. If this score is large, then it means that the corresponding feature has a significant impact on the transition. We provided a detailed interpretation of this feature importance score in the application which

provides insights for disease progression diagnosis for clinical trial data in Section 5.6.1.

5.5 Simulations

5.5.1 Test the prediction performance

We generated time series from a Gaussian HMM with 2 hidden states and 10 cross predictors, where the emission probabilities associated with state i are distributed as $N(\mu_i, \sigma^2 I)$ where $\mu_1 = [1, 0, \cdots, 0]^\top$, $\mu_2 = [0, 1, 0, \cdots, 0]^\top$, and I is the $\dim(Y) \times \dim(Y)$ identity matrix. For the transition probabilities, we use the multinomial-logistic form stated in Equation 5.5. We generated cross predictors of dimensionality $\dim(X)$ *i.i.d.* from $N(0, 1)$. Then we sampled the coefficients β_{rs}^m's from *i.i.d.* $Unif[10, 20] \times \delta_{rs}^m$, where $\delta_{rs}^m \sim Bern(p), p = \{0.01, 0.05, 0.1, 0.5, 1.0\}$ to control the sparsity of the cross predictor. In other words, if $\delta_{rs}^m = 0$ we let $\beta_{rs}^m = 0$ otherwise $\beta_{rs}^m \sim Unif[10, 20]$. We simulated data across a wide range of settings: $\sigma = \{0.01, 0.05, 0.1, 0.5\}$; $\dim(X) = 10, 100$; $\dim(Y) = 10, 100$. For each setting, we repeated our simulation 100 times and reported the average R^2 and computational time. We used 10000 time points for training and tested the performance on a test series of size 1000. We test the performance on feature importance inference of this latent control model, and also compare its prediction results with control HMM as well as the oracle using all information up to time t including h_t and all parameters to predict $t + 1$.

We compared the prediction performance of latent control HMM with some benchmarks such as control HMM and the oracle. From Figure 5.4 we can see that the

latent control HMM performs much better than other models, and it even nearly overlaps with the oracle under different signal-noise ratio.

5.5.2 Test the feature selection

Here we tested whether the latent control HMM models can select the true important features. We generate data similar to Section 5.5.1 but with different sparsity ratio, including $0.7, 0.8, 0.9$. Since there are 2 hidden states and 10 cross predictors, in total there are 40 cross predictor coefficients. If the sparsity ratio is 0.7 / 0.8 / 0.9, then 28 / 32 / 36 among 40 coefficients are 0. Then we conducted MCMC sampling 500 times, so for each Z_{rs}^m, we have 500 samples and we can calculate their feature importance by their posterior mean. We repeat the whole experiment 10 times, and then calculate the average probability that a feature could be selected in the model condition on it is a true important feature, and the average probability that a feature could be selected in the model condition on it is not a true important feature. Therefore, the larger the difference between these two probabilities, the better feature selection ability the model will have. Figure 5.5 shows the results of this experiment. We can see that the feature importance is separable, that the strong features have a much higher probability to have the corresponding binary control variable being 1. Here we can see when the model is sparser, then the difference between the two probabilities are larger, which indicates that the model feature selection ability is stronger. Compared with control HMM under E-M framework, this model incorporates all cross predictors, therefore the above two probabilities are always 1. It suggests that our method works better and can identify the important features under sparse cases.

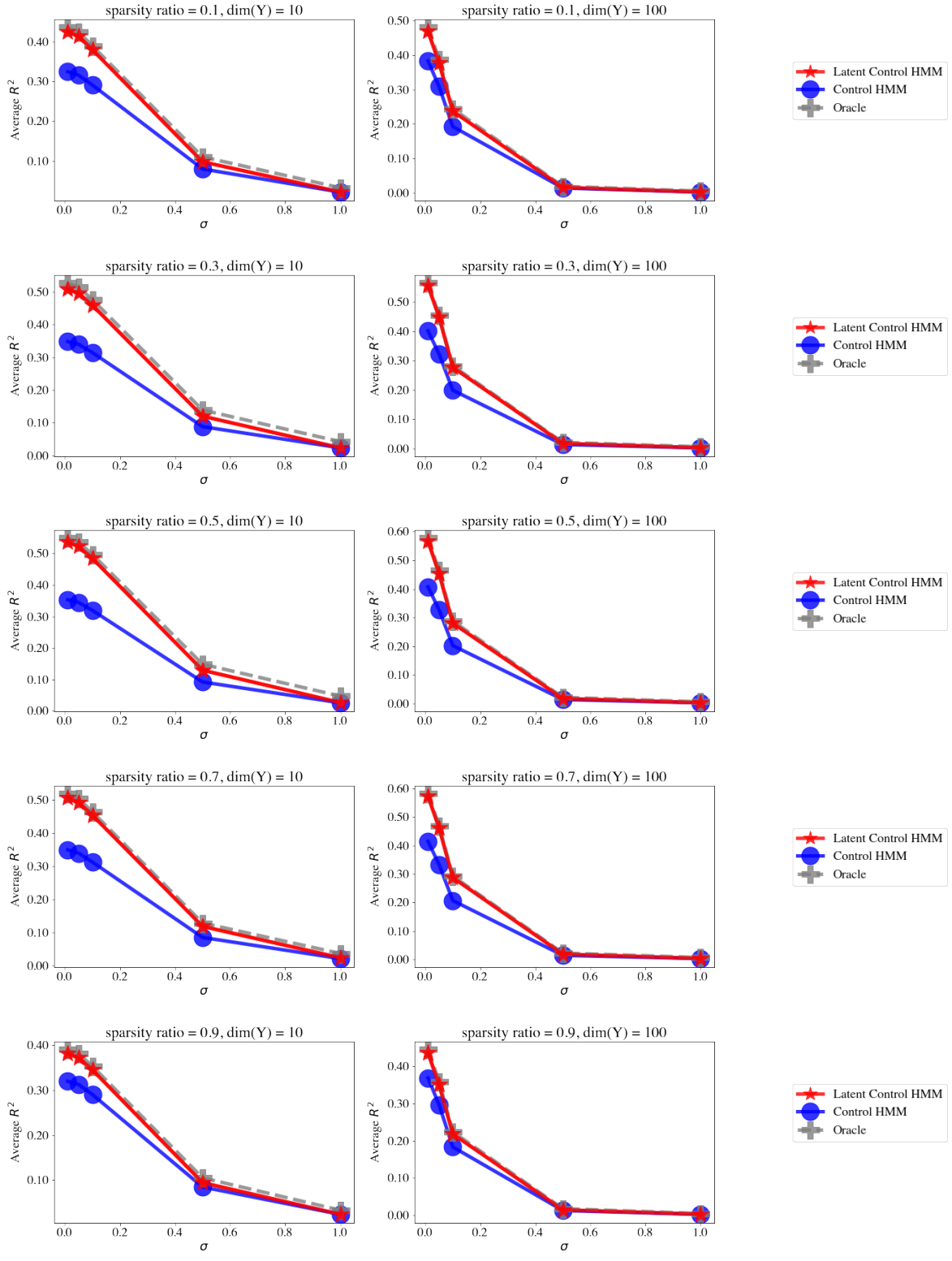


Figure 5.4: This figure shows the simulation results for the prediction performance of latent control model and the benchmarks under different signal noise ratio and sparsity of coefficient cases. The left column shows the results of low dimensional data, and the right column shows high dimensional cases.

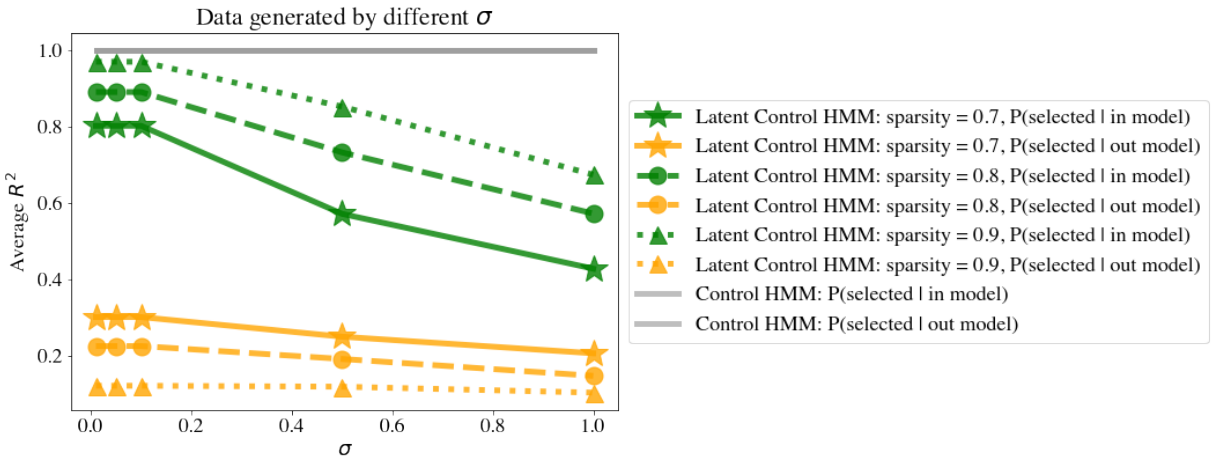


Figure 5.5: This figure shows the probabilities of a strong or noisy feature selected in the model.

5.6 Application

We show two applications of this latent control HMM model. The first one is to show the inference and feature selection ability of this model, and the second application is to show the strong prediction performance of this model.

5.6.1 Application I: Inference of latent control on end-stage gastric cancer data

This application is to use latent control HMM with multiple observed series. The data was extracted from patients' medical reports (1000+ pages) provided by University of Virginia Health System (UVA Health), including the lab reports, scan reports and the pathology reports. There are 96 patients and 10 variables included in this dataset. Those variables contain information of patients' demographic characteristics and lab results. The six lab results are collected weekly during their chemotherapy. Among

those variables, the 4 demographic characteristics are the exogenous predictors, the lab results are response, and each patient forms an observed series with time-invariant cross predictors. The following Table 5.1 shows more detailed description of the variables. We inferred it with a latent control Gaussian HMM with 2 states and 1000 MCMC samples. All exogenous are standardized to have 0 mean and 1 variance.

	Variables	Description	Data type & Normal range	Unit
Demographic (at the time of diagnosis)	dx-age	Patient's age	integer	
	diabetes	Diabetic or not	binary	
	gender	Gender of patient	binary	
	BIM	Body mass index	continuous	$kg/m2$
Lab results	T_1	Hematocrit	continuous, 35.0-47.0	%
	T_2	Platelet	continuous, 150-450	K/UL
	T_3	WBC	continuous, 4.0-11.0	K/UL
	T_4	Albumin	continuous, 3.2-5.2	G/DL
	T_5	Creatinine	continuous, 0.7-1.3	MG/DL
	T_6	Total bilirubin	continuous, 0.3-1.2	MG/DL

Table 5.1: Variables in end-stage gastric cancer dataset. Data was collected from UVA Health.

Results and interpretation Table 5.2 shows the estimated mean for each cluster, i.e. the estimated mean for different emission distributions. According to the reference level and normal range of the lab results in Table 5.1, we labeled the abnormal lab results with stars ('*') in Table 5.2. With this table, we could interpret the two hidden states as 'faster progression state' and 'lower progression state' respectively according to the lab results. That is to say, if an inferred state has more abnormal estimated mean of lab results, then we treat it as 'faster progression state'; otherwise, we labeled it as 'lower progression state'. Therefore, the transition from 'lower progression state' to 'faster progression state' indicates the patient's lab results become worse and have a severe condition than the last observation. In contrast, the transition from 'faster

progression state' to 'lower progression state' indicates the patient's lab results become better but this doesn't indicate that the patient has fully recovered. As we can see, in this table, 5 out of 6 estimated means of the lab results in 'faster progression state' are out of the reference level, while no estimated means are out of the reference range in the 'lower progression state'. We also checked the estimated standard deviation, which showed that 'faster progression state' has a much larger value than 'lower progression state'. Note that 'lower progression' and 'faster progression' are relative concepts, since all patients are end-stage gastric cancer patients and died in the hospitals or hospices eventually.

Cluster	$T1$	$T2$	$T3$	$T4$	$T5$	$T6$
Reference	(35.0, 47.0)	(150, 450)	(4.0, 11.0)	(3.2, 5.2)	(0.7, 1.3)	(0.3, 1.2)
Faster progression	27.42*	244.62	11.51*	2.64*	1.33*	1.21*
Lower progression	33.18	275.88	8.29	3.36	0.78	0.47

Table 5.2: Estimated mean for each cluster in gastric cancer dataset.

		h_{t+1}	
		Faster	Lower
h_t	Faster	0.216 +	0.571 +
	Lower	0.703 +	0.542 +

(a) Age at diagnosis

		h_{t+1}	
		Faster	Lower
h_t	Faster	0.56 +	0.254 +
	Lower	0.483 -	0.41 +

(b) Gender (0: male, 1: female)

		h_{t+1}	
		Faster	Lower
h_t	Faster	0.891 +	0.715 -
	Lower	0.461 +	0.518 -

(c) Diabetes (0: N, 1: Y)

		h_{t+1}	
		Faster	Lower
h_t	Faster	0.675 -	0.399 +
	Lower	0.233 +	0.31 +

(d) Body mass index.

Table 5.3: The probability of an exogenous predictor controlling the transition among hidden states.

Table 5.3 shows the estimated probability that a cross predictor controlling the tran-

sition between hidden states. For example, '0.703 +' in Table 5.3 can be interpreted as: the age at time of diagnosis can influence the hidden states transiting from 'lower progression' to 'faster progression' with probability 0.703. i.e. among every 100 MCMC samples, about 70 samples shows that the age at time of diagnosis can influence the hidden states transiting from 'lower progression' to 'faster progression'. '+' means the impact is positive (i.e. $\beta^{dx-age}_{lower,faster} > 0$), which indicated that the larger age the patient has, the more likely their state will transfer from 'lower progression' to 'faster progression' that the condition will become worse. From the results shown in Table 5.3, we can see that patients who are elder, diabetic and with a smaller BMI are more likely to have a severe condition. This is consistent with some prior knowledge in clinical diagnosis. For example, end-stage gastric cancer patients who have a larger BMI indicated that those patients have a better condition than others, since the tumor development usually yields losing weight dramatically (Tegels et al., 2014).

5.6.2 Application II: Prediction of natural gas's volatility

This application is to use latent control HMM with time-dependent exogenous predictors. The data of natural gas future contract price (2000.01 to 2020.05) was downloaded from Yahoo finance, and New York weather data was downloaded from National Weather Service. Table 5.4 shows the variables used in the dataset. In this application, we use volatility as the response, and use the temperature as time-dependent exogenous predictors. Here we used the difference between the highest price and lowest price of each day to describe the volatility of the natural gas' price. The volatility is not tradable directly, but we can use the result as a signal or indicator to help option trading, such as Gamma trading or straddles. We inferred it

with a latent control Gaussian HMM with 3 states and 1000 MCMC samples. All exogenous are standardized to have 0 mean and 1 variance. We compared it with the benchmark method, control HMM inferred by MLE.

	Variables	Description	Unit
raw	Date	the date that the data recorded	
variables	High	the highest price of the NG future contract	USD
	Low	the lowest price of the NG future contract	USD
	TMAX	the highest temperature of NYC	$°C$
	TMIN	the lowest temperature of NYC	$°C$
response	Vol	volatility = High - Low	USD
exogenous	THIGH	binary, 1 if $TMAX > 30$, else 0	
predictors	TLOW	binary, 1 if $TMIN < 0$, else 0	

Table 5.4: Variables in natural gas prediction dataset

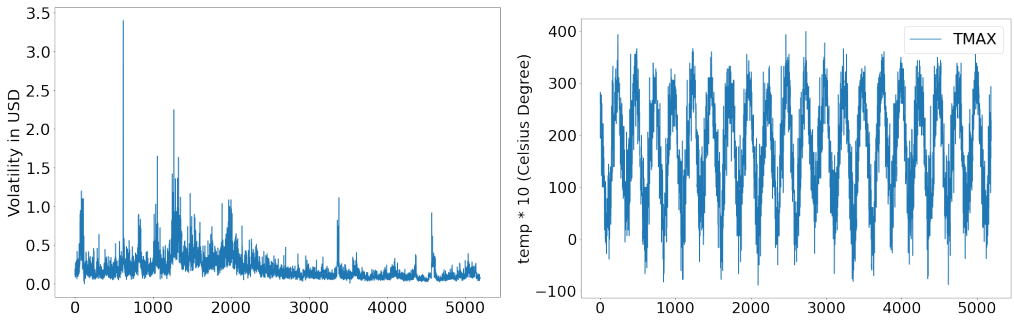


Figure 5.6: Time series plot

Results and interpretation. We tested the latent control HMM on this natural gas contracts data set to predict the volatility of the price, compared the results with the benchmark method, control HMM, and the underlying ground truth. Table 5.5 shows the results of prediction. From the table, we can see that R^2 for latent variable HMM is 0.32, R^2 for EM-based control HMM is 0.26. Using 0.2 as a threshold for classifying the volatility, above which the volatility is classified into high volatility, otherwise, low volatility, then we calculated Area under the Receiver Operating Characteristic Curve (ROC-AUC) based on this classification. The results show that latent

control HMM outperformed control HMM. Based on MCMC samples, we constructed the 95% Bayesian credible interval shown in Figure 5.7.

Model	Inference	Prior	test R^2	AUC
Latent control HMM	MCMC	with prior	0.33	0.912
control HMM	MLE	no prior	0.26	0.904

Table 5.5: R squared of the prediction of natural gas

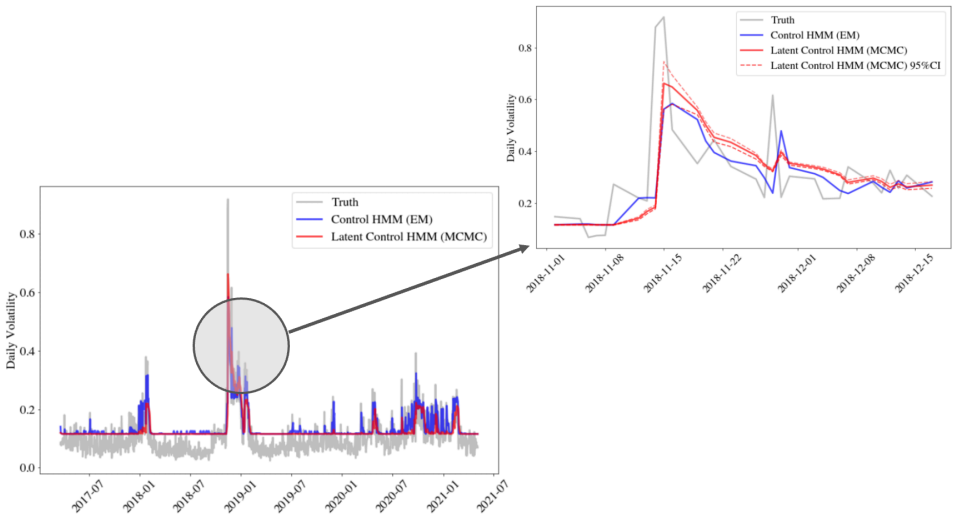


Figure 5.7: Compare the prediction by latent control HMM with control HMM, the latent control HMM gives a higher R^2.

5.7 Discussions

There are two main advantages of latent control HMM, the predictability and feature importance interpretability under sparse settings. Both two advantages come from the MCMC sampling procedure because for each feature it is taking both two possibilities, in the model or out of the model, into consideration. From MCMC sampling, we will have the posterior of whether each feature is involved in the model, which will be automatically calibrated according to the observed data.

The latent control variable could provide an interpretation of how the exogenous predictors can impact the transition between the hidden states, which could provide some insights and signals in real-world applications, such as for disease progression diagnosis. The feature importance provided by latent control is unique because it is a marginal posterior probability. In contrast, Bayesian HMM proposed by Shirley et al. (2010) cannot provide feature importance. Control HMM can evaluate the feature importance by Likelihood Ratio Test (LRT) (Woolf, 1957) under frequentist framework, but it has two drawbacks. First, it provides a conditional, not marginal, p-value. The LRT is comparing two nested models, and the p-value is only valid conditioned on the smaller model. Second, it provides a p-value indicating whether a feature is significant or not. This is an in the model or out of model qualitative interpretation, instead of a quantitative posterior probability interpretation.

The latent control can help prediction because it provides flexibility of incorporation of the cross features. The binary latent control indicates whether the model involves the features or not, which can be told by the posterior. This would be useful when we are not confident of whether to involve the feature or not and the prediction is a Bayesian model averaging. In contrast, control HMM will always involve all cross predictors. Besides, control HMM is inferred by MLE so will only converge to the local optima instead. But our latent control HMM samples from posterior distribution which will be less likely to be trapped into local optima.

Appendices

Appendix A

Appendix

A.1 Detailed Simulation Results for PSHMM and Online Learning Variants

Table A.1 shows the detailed simulation results for Figure 3.2.

Table A.2 shows the detailed simulation results for Figure 4.1.

E	T	σ	#Cluster inferred	Limited oracle	Strong oracle	SHMM	PSHMM simplex	PSHMM polyhedron
t_5	sticky	0.05	5	0.28	0.28	−673629.23	0.27	0.24
t_{10}	sticky	0.05	5	0.3	0.3	−67.72	0.29	0.27
t_{15}	sticky	0.05	5	0.31	0.31	−2.82	0.3	0.28
t_{20}	sticky	0.05	5	0.3	0.3	−205.24	0.29	0.27
t_5	nonsticky	0.05	5	0.17	0.17	−2.48	0.17	0.11
t_{10}	nonsticky	0.05	5	0.19	0.19	0.11	0.18	0.11
t_{15}	nonsticky	0.05	5	0.19	0.19	−0.59	0.18	0.11
t_{20}	nonsticky	0.05	5	0.19	0.19	0.14	0.19	0.11
N	sticky	0.05	3	0.31	0.31	−125.03	0.21	0.12
N	sticky	0.05	4	0.31	0.31	−717.8	0.25	0.23
N	nonsticky	0.05	3	0.19	0.19	−3.25	0.16	0.1
N	nonsticky	0.05	4	0.19	0.19	−155.85	0.17	0.09
N	sticky	0.01	5	0.38	0.38	0.34	0.38	0.37
N	nonsticky	0.01	5	0.24	0.24	0.24	0.24	0.15
N	sticky	0.05	5	0.31	0.31	−9.36	0.3	0.28
N	nonsticky	0.05	5	0.19	0.19	0.14	0.19	0.12
N	sticky	0.1	5	0.19	0.19	−74.73	0.18	0.15
N	sticky	0.5	5	0.01	0.01	−22.02	0.01	0.0
N	sticky	1.0	5	0.0	0.0	−1163.8	0.0	−0.0
N	nonsticky	0.1	5	0.12	0.12	−8.47	0.12	0.07
N	nonsticky	0.5	5	0.01	0.01	−22.29	0.01	0.0
N	nonsticky	1.0	5	0.0	0.0	−5.61	0.0	−0.0

Table A.1: Detailed simulation results for Figure 3.2.

E	T	σ	#Cluster inferred	PSHMM simplex	PSHMM simplex, online	PSHMM simplex, online, decay
t_5	sticky	0.05	5	0.27	0.26	0.07
t_{10}	sticky	0.05	5	0.29	0.29	0.07
t_{15}	sticky	0.05	5	0.3	0.29	0.06
t_{20}	sticky	0.05	5	0.29	0.29	0.06
t_5	nonsticky	0.05	5	0.17	0.17	−0.02
t_{10}	nonsticky	0.05	5	0.18	0.18	−0.02
t_{15}	nonsticky	0.05	5	0.18	0.18	−0.03
t_{20}	nonsticky	0.05	5	0.19	0.18	−0.02
N	sticky	0.05	3	0.21	0.21	0.1
N	sticky	0.05	4	0.25	0.25	0.08
N	nonsticky	0.05	3	0.16	0.17	0.04
N	nonsticky	0.05	4	0.17	0.18	−0.0
N	sticky	0.01	5	0.38	0.37	0.08
N	nonsticky	0.01	5	0.24	0.24	−0.08
N	sticky	0.05	5	0.3	0.3	0.06
N	nonsticky	0.05	5	0.19	0.19	−0.03
N	sticky	0.1	5	0.18	0.18	0.03
N	sticky	0.5	5	0.01	0.0	−0.0
N	sticky	1.0	5	0.0	−0.01	−0.0
N	nonsticky	0.1	5	0.12	0.11	0.0
N	nonsticky	0.5	5	0.01	−0.0	−0.01
N	nonsticky	1.0	5	0.0	−0.01	−0.01

Table A.2: Detailed simulation results for Figure 4.1.

Bibliography

Anandkumar, Anima et al. (2012). "A spectral algorithm for latent dirichlet allocation". In: *Advances in neural information processing systems* 25.

Anandkumar, Animashree et al. (2011). "Spectral methods for learning multivariate latent tree structure". In: *Advances in neural information processing systems* 24.

Andrieu, Christophe et al. (2003). "An introduction to MCMC for machine learning". In: *Machine learning* 50.1, pp. 5–43.

Baum, Leonard E and John Alonzo Eagon (1967). "An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology". In: *Bulletin of the American Mathematical Society* 73.3, pp. 360–363.

Baum, Leonard E and Ted Petrie (1966). "Statistical inference for probabilistic functions of finite state Markov chains". In: *The annals of mathematical statistics* 37.6, pp. 1554–1563.

Baum, Leonard E, Ted Petrie, et al. (1970). "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains". In: *The annals of mathematical statistics* 41.1, pp. 164–171.

Bengio, Yoshua and Paolo Frasconi (1994). "An input output HMM architecture". In: *Advances in neural information processing systems* 7.

Bhar, Ramaprasad and Shigeyuki Hamori (2004). *Hidden Markov models: Applications to financial economics*. Vol. 40. Springer Science & Business Media.

Boyd, Stephen, Stephen P Boyd, and Lieven Vandenberghe (2004). *Convex optimization*. Cambridge university press.

Chekhlov, Alexei, Stanislav Uryasev, and Michael Zabarankin (2004). "Portfolio optimization with drawdown constraints". In: *Supply chain and finance*. World Scientific, pp. 209–228.

— (2005). "Drawdown measure in portfolio optimization". In: *International Journal of Theoretical and Applied Finance* 8.01, pp. 13–58.

Chen, Yen-Chi (2022). "Statistical inference with local optima". In: *Journal of the American Statistical Association*, pp. 1–13.

Chib, Siddhartha and Edward Greenberg (1995). "Understanding the metropolis-hastings algorithm". In: *The american statistician* 49.4, pp. 327–335.

Dempster, Arthur P, Nan M Laird, and Donald B Rubin (1977). "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1, pp. 1–22.

Eberly, Lynn E and George Casella (2003). "Estimating Bayesian credible intervals". In: *Journal of statistical planning and inference* 112.1-2, pp. 115–132.

Eckart, Carl and Gale Young (1936). "The approximation of one matrix by another of lower rank". In: *Psychometrika* 1.3, pp. 211–218.

Falck, Antoine, Adam Rej, and David Thesmar (2022). "When do systematic strategies decay?" In: *Quantitative Finance* 22.11, pp. 1955–1969.

Fama, Eugene F and Kenneth R French (2012). "Size, value, and momentum in international stock returns". In: *Journal of financial economics* 105.3, pp. 457–472.

Fontenla-Romero, Óscar et al. (2013). "Online machine learning". In: *Efficiency and Scalability Methods for Computational Intellect*. IGI global, pp. 27–54.

Frisch, KR (1955). "The logarithmic potential method of convex programming". In: *Memorandum, University Institute of Economics, Oslo* 5.6.

Goldberg, Lisa R and Ola Mahmoud (2017). "Drawdown: from practice to theory and back again". In: *Mathematics and Financial Economics* 11, pp. 275–297.

Grossman, Sanford J and Zhongquan Zhou (1993). "Optimal investment strategies for controlling drawdowns". In: *Mathematical finance* 3.3, pp. 241–276.

Halko, Nathan, Per-Gunnar Martinsson, and Joel A Tropp (2011). "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions". In: *SIAM review* 53.2, pp. 217–288.

Harris, Charles R. et al. (Sept. 2020). "Array programming with NumPy". In: *Nature* 585.7825, pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

Hsu, Daniel and Sham M Kakade (2013). "Learning mixtures of spherical gaussians: moment methods and spectral decompositions". In: *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pp. 11–20.

Hsu, Daniel, Sham M Kakade, and Tong Zhang (2012). "A spectral algorithm for learning hidden Markov models". In: *Journal of Computer and System Sciences* 78.5, pp. 1460–1480.

Ishwaran, Hemant and J Sunil Rao (2005a). "Spike and slab variable selection: frequentist and Bayesian strategies". In.

— (2005b). "Spike and slab variable selection: frequentist and Bayesian strategies". In: *The Annals of Statistics* 33.2, pp. 730–773.

Jaeger, Herbert (2000). "Observable operator models for discrete stochastic time series". In: *Neural computation* 12.6, pp. 1371–1398.

Jamshidian, Mortaza and Robert I Jennrich (2000). "Standard errors for EM estimation". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 62.2, pp. 257–270.

Lahmiri, Salim and Stelios Bekiros (2021). "Deep learning forecasting in cryptocurrency high-frequency trading". In: *Cognitive Computation* 13, pp. 485–487.

Li, Ren-Cang (2006). "Matrix perturbation theory". In: *Handbook of linear algebra*. Chapman and Hall/CRC, pp. 15–1.

Mamon, Rogemar S and Robert J Elliott (1995). *Hidden Markov Models in Finance: Further Developments and Applications, Volume II*. Springer.

McLachlan, Geoffrey J and Kaye E Basford (1988). *Mixture models: Inference and applications to clustering*. Vol. 38. M. Dekker New York.

McQueen, Grant (1992). "Long-horizon mean-reverting stock prices revisited". In: *Journal of Financial and Quantitative Analysis* 27.1, pp. 1–18.

Needle, Sagl B and Christus D Wunsch (1970). "A general method applicable to the search for similarities in the amino acid sequence of two proteins". In: *Journal of molecular biology* 48.3, pp. 443–53.

Pearson, Karl (1936). "Method of moments and method of maximum likelihood". In: *Biometrika* 28.1/2, pp. 34–59.

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Rodu, Jordan (2014). *Spectral estimation of hidden Markov models*. University of Pennsylvania.

Rodu, Jordan et al. (2013). "Using regression for spectral estimation of hmms". In: *International Conference on Statistical Language and Speech Processing*. Springer, pp. 212–223.

Rydén, Tobias (2008). "EM versus Markov chain Monte Carlo for estimation of hidden Markov models: A computational perspective". In: *Bayesian Analysis* 3.4, pp. 659–688.

Sankoff, David (1972). “Matching sequences under deletion/insertion constraints”. In: *Proceedings of the National Academy of Sciences* 69.1, pp. 4–6.

Sharpe, William F (1966). “Mutual fund performance”. In: *The Journal of business* 39.1, pp. 119–138.

Shirley, Kenneth E et al. (2010). “Hidden Markov Models for Alcoholism Treatment Trial Data”. In: *The Annals of Applied Statistics* 4.1, p. 366.

Tegels, Juul JW et al. (2014). “Improving the outcomes in gastric cancer surgery”. In: *World journal of gastroenterology: WJG* 20.38, p. 13692.

Virtanen, Pauli et al. (2020). “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17, pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

Viterbi, Andrew (1967). “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm”. In: *IEEE transactions on Information Theory* 13.2, pp. 260–269.

Wang, Weiran and Miguel A Carreira-Perpinán (2013). “Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application”. In: *arXiv preprint arXiv:1309.1541*.

Woolf, Barnet (1957). “The log likelihood ratio test (the G-test)”. In: *Annals of human genetics* 21.4, pp. 397–409.

Yang, Fanny, Sivaraman Balakrishnan, and Martin J Wainwright (2017). “Statistical and computational guarantees for the Baum-Welch algorithm”. In: *The Journal of Machine Learning Research* 18.1, pp. 4528–4580.