A Novel Multivariate Hidden Markov Model Learning method using Coupled Canonical Polyadic Decomposition

Applied on the Sleep Physionet dataset to uncover sleep stages

Jep Brinkmann

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

A Novel Multivariate Hidden Markov Model Learning method using Coupled Canonical Polyadic Decomposition

Applied on the Sleep Physionet dataset to uncover sleep stages

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Jep Brinkmann

Dr.ir. Kim Batselier
Dr.ir. Borbála (Bori) Hunyadi

June 20, 2025

Faculty of Mechanical Engineering (ME) · Delft University of Technology

The image used for the cover page background was created using Leonardo.ai

Abstract

Hidden Markov Models (HMMs) are probabilistic models that are widely used in various fields, including machine learning, economics, information theory, neuroimaging, and more. In particular, they are frequently employed in functional Magnetic Resonance Imaging (fMRI) studies, which involve large datasets, to analyze the behavior of brain networks or states. For example, HMMs can help determine whether a patient is healthy.

HMMs represent the probability of transitions between different states of a system, which operates as a discrete Markov chain and is not directly observable. Additionally, they describe the probability of observing specific measurements based on the current state of the model. These probabilities are characterized by a state transition matrix ($\mathbf{T}$), an emission matrix ($\mathbf{O}$), and an initial state distribution ($\boldsymbol{\pi}$).

The current methods for fitting HMMs to data primarily utilize the Baum-Welch algorithm, which is a specific type of Expectation Maximization (EM). One of the advantages of Baum-Welch is its ability to be extended to continuous observations or multivariate data. However, the algorithm involves a forward-backward pass during each iteration, which makes it increasingly slow as the size of the dataset grows. Moreover, it can easily become trapped in local minima.

An alternative approach is to use Canonical Polyadic Decomposition (CPD) to decompose a Joint Probability Tensor (JPT). This decomposition allows for the extraction of factor matrices that can be used to calculate the HMM matrices ($\mathbf{T}, \mathbf{O}, \boldsymbol{\pi}$). Compared to the Baum-Welch algorithm, CPD and other JPT decomposition methods tend to be faster. However, they are often sensitive and may suffer from instability if the data does not fully capture the statistical behavior of the underlying HMM. Additionally, methods for JPT decomposition in HMM learning have not yet been extended to multivariate datasets or continuous settings.

To enhance stability and accommodate multivariate data, we propose a novel method that extends JPT decomposition HMM learning to a multivariate setting. This involves using a coupled CPD problem, where each observational sequence has a separate observation matrix but shares a common state transition matrix. By transitioning from Baum-Welch to CPD, the process of learning HMMs can be significantly accelerated. Coupled CPD makes HMM

learning more robust than Uncoupled CPD by relating the multivariate data through a common transition matrix $\mathbf{T}$ and initial distribution $\boldsymbol{\pi}$. This improvement should enhance the iterability of HMM methods and make them more suitable for rapid prototyping in scientific research and the aforementioned fields.

We show in the results that Coupled CPD indeed outperforms both Uncoupled CPD and the industry-standard Baum-Welch, in most cases. It has improved stability over both methods, as well as significantly improving the calculation time, and results in higher accuracy when it comes to estimating the HMM matrices. Finally, more future improvements are suggested concerning calculation time and extensions.

Table of Contents

Acknowledgements

I would like to express my gratitude to my supervisors Kim Batselier and Bori Hunyadi. They have been very understanding, supportive, critical in a good way, and overall very pleasant to work with. I started working with them under unusual circumstances, and am now finishing up with a feeling of pride and excitement about the work I've done, which is not something I would've had without them as my supervisors.

I also would like to thank my family and my friends from Systems and Control who made my time at TU Delft all the better. There were early mornings and late nights studying and discussing the material. Through those conversations and discussions, I have learned more than all the lectures combined. Next to learning from them academically, I have also learned a lot from them personally, and am grateful to be included in such a warm and welcoming group. I hope that our 50-year anniversary will just be a regular Tuesday.

Special thanks go out to Maria de Fonseca and Tatsuki Fujioka, for they have helped me tremendously throughout my thesis process with feedback and debate. At this point, Maria probably knows my report better than I do, and Tatsuki knows my method better than I do. Next to that, they have supported me outside of just the work as well, and incredibly so.

Delft, University of Technology — Jep Brinkmann
June 20, 2025

Notation

This section introduces a general use of commonly used notation throughout the report. Here, symbols, variable types, and index uses can be found. Note that this is a general description, and locally throughout the report, this may vary.

Table 1: Notation used in this report.

Representation	**Definition**
$\underline{\mathbf{X}}$	Tensor
$\mathbf{X}$	Matrix
$\mathbf{x}$	Vector
x	Scalar
$\mathbf{X}^{(n)}$	n-th matrix in a sequence of matrices $\left(\left\{\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(n)}\right\}\right)$
$\mathbf{X}_{(n)}$	mode- n matricization of a tensor X
Operations	**Definition**
$\circ$	Outer product
$\odot$	Khatri-Rao
$\circledast$	Hadamard product
$\times_n$	Mode-n tensor-matrix product
$\mathbf{A}^{-1}$	Matrix inverse
$\mathbf{A}^{\dagger}$	Matrix pseudo-inverse
$\mathbf{A}^{T}$	Matrix transpose
$\|\cdot\|_F$	Frobenius norm
Variables	**Definition**
K	Number of states of HMM[1]
D	Number of possible emissions (data bins) of the HMM
N	Number of data sequences of multivariate HMM
m	Length of the data sequence for HMM fitting
W	Window size of indicator function/dimension-size of JPT[2].
$k \in [1, K]$	Indexing variable for the number of states
$d \in [1, D]$	Indexing variable for the number of emissions

[1] Hidden Markov Model (HMM)
[2] Joint Probability Tensor (JPT)

$n \in [1, N]$	Indexing variable for the number of sequences

The notation for Joint Probability Tensors will be as follows: ${}^{m}\underline{\mathbf{P}}^{n}_{a,b,c}$, where m is the number of samples used to create the tensor, where ${}^{\infty}\underline{\mathbf{P}}$ indicates a theoretically calculated tensor instead of an empirical sampled one equating to infinite samples, n is the indexing variable for which data sequence is used in the case of a multivariate HMM, and a, b, c are index variables pointing to a specific element of the tensor, and the number of them can refer back to the window size/dimension size of the tensor. $\underline{\mathbf{P}}$ will be used if all are unspecified.

Sometimes the notation for multivariate emission matrices $\mathbf{O}^{(n)} \quad \forall n \in [1, N]$ is shortened to $\mathbf{O}^{(n)}$.

Chapter 1

Introduction

Hidden Markov Models (HMMs) are probabilistic models that are widely used in various fields, including machine learning, economics, information theory, neuroimaging, and more. In particular, they are frequently employed in functional Magnetic Resonance Imaging (fMRI) and Electroencephalography (EEG) studies, which involve large datasets, to analyze the behavior of brain networks or states, and determine whether the patient is healthy or not [1, 2, 3]. HMMs represent observed sequences as emissions from a set of underlying hidden states and are defined by three key parameters: the transition matrix $\mathbf{T}$, the emission matrix $\mathbf{O}$, and the initial state distribution $\boldsymbol{\pi}$.

Learning these parameters from data poses a significant challenge [4]. The most common method found in the literature is named the Baum-Welch algorithm [4, 5, 6], a specific implementation of the Expectation Maximization (EM) technique. This algorithm performs a forward-backward pass through the data at each iteration. While it is effective, Baum-Welch has limitations, as it tends to scale poorly with larger datasets and is susceptible to converging on local minima. An alternative approach, tensor decomposition—particularly the Canonical Polyadic Decomposition (CPD)—offers a promising but less thoroughly explored method for estimating HMM parameters [6, 7, 8]. However, CPD-based methods face several limitations [9]. They have undergone limited experimental validation, may experience sensitivity issues, and crucially, have not yet been extended to multivariate cases. In these scenarios, multiple observed sequences are assumed to share the same underlying hidden states over time, with different observation models for each sequence. These issues naturally lead to the central research question of this thesis:

Can CPD methods for HMM learning be improved to accommodate multivariate data sequences while outperforming the industry-standard, Baum-Welch algorithm, in both computational efficiency and estimation accuracy?

To address this gap, this thesis aims to extend univariate CPD-based HMM learning methods (Uncoupled CPD) to the multivariate setting. In this context, several additional research questions arise:

1. How can CPD methods for HMM learning decompose Joint Probability Tensors (JPTs) to derive HMMs?
2. How can CPD methods for HMM learning be extended to a multivariate setting?
3. How can CPD methods for HMM learning be extended to increase stability and robustness?
4. Can robustness and the multivariate setting be achieved by coupling data sequences using a common transition matrix $\mathbf{T}$ and initial distribution π?
5. How does a Coupled CPD method compare in performance to Uncoupled CPD and the Baum-Welch algorithm over various HMM parameters, such as the number of states K, the number of emissions D, the number of data sequences N, and the length of the data sequences m?
6. Can the Coupled CPD method be applied to the Sleep Physionet dataset [10] to uncover underlying sleep stages?

These questions are addressed throughout this thesis in different chapters. Question 1 is discussed in Chapter 2, while questions 1-4 are addressed in Chapter 3. Finally, questions 4-6 are explored in Chapter 4.

The proposed approach builds on the observation that all sequences in a multivariate HMM share the same transition matrix $\mathbf{T}$ and initial distribution $\boldsymbol{\pi}$, while allowing for distinct emission matrices $\mathbf{O}^{(n)}$ for each sequence. This extension aims to combine the computational efficiency of CPD methods with the multivariate capabilities of the Baum-Welch algorithm while simultaneously improving accuracy.

To test the proposed framework in a real-life setting, the Sleep Physionet dataset, including polysomnography data and annotations for different sleep stages from [10], is used. The goal is to use the EEG and EOG channels from this dataset to estimate the underlying sleep stages during the night and compare these to the sleep stage annotations given by experts for the same dataset.

The remainder of the thesis is structured as follows: Chapter 2 introduces relevant background concepts and reviews existing literature. Chapter 3 provides a comprehensive description of univariate HMM learning using CPD, based on existing methods, and introduces a novel extension to the multivariate case by coupling multiple univariate models. This chapter also discusses implementation considerations and defines auxiliary functions. Chapter 4 evaluates the proposed method through experiments that explore a wide range of HMM parameters and method-specific hyperparameters. Finally, Chapter 5 summarizes the findings, concludes the thesis, and proposes directions for future work. Supplementary materials are provided in Appendix A, which presents algorithmic details, Appendix B, which contains supporting mathematical proofs, and Appendix C, which supplies extra supporting figures.

Chapter 2

Preliminaries and Related Work

This chapter describes the basis for the proposed method of this thesis. The goal is to learn Hidden Markov Models (HMMs) from data in a multivariate setting through coupled Canonical Polyadic Decomposition (CPD). To do so, first, HMMs are defined in Section 2-1. Then in Section 2-2, Joint Probability Tensors (JPTs) are shown to be an intermediate step that can be obtained from data, and decomposed to obtain the HMM matrices. Finally, in Section 2-3 the method of decomposition of this JPT into HMM matrices is described. Throughout these sections, various related works are discussed, some of which are used as a basis for this work.

2-1 Hidden Markov Models

A discrete univariate Hidden Markov Model (HMM) is a probabilistic model defined by a transition matrix $\mathbf{T} \in \mathbb{R}^{K \times K}$, and an emission matrix $\mathbf{O} \in \mathbb{R}^{D \times K}$. These describe the probabilistic evolution of the hidden states, $x_t \in \mathcal{X} = \{s_1, s_2, \ldots, s_K\}$, and the probability of observing one of the possible observational values $y_t \in \mathcal{Y}$ from the alphabet $\mathcal{Y} = \{o_1, o_2, \ldots, o_D\}$. Lastly, an HMM also has an initial state distribution vector $\boldsymbol{\pi} \in \mathbb{R}^K$, which describes the probability of starting a measured sequence in a certain state.

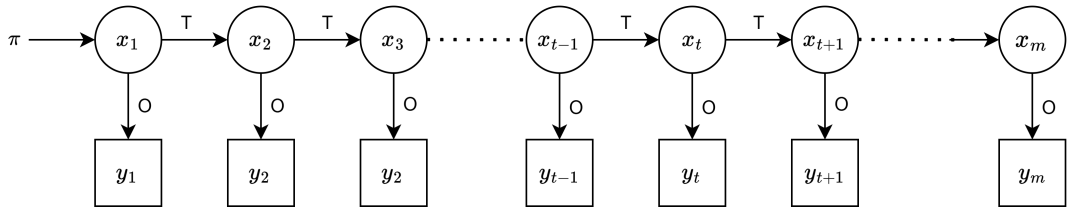


Figure 2-1: An illustration of an HMM time sequence with observed data and the underlying states.

An HMM describes a sequence of observations $y_t \in \{y_1, y_2, \ldots, y_m\}$, for each timestep $1 \leq t \leq m$, where m is the length of the data sequence, which depends on a sequence of hidden states

$x_t \in \{x_1, x_2, \ldots, x_m\}$, as can be observed in Figure 2-1. Each observation at time t belongs to a set of possible observational values $y_t \in \mathcal{Y}$, as does each of the hidden states $x_t \in \mathcal{X}$.

The probability of moving from state to state is described by the state transition probability matrix $\mathbf{T}$, and the probability of observing a specific value given a specific state is described by the emission probability matrix $\mathbf{O}$. Furthermore, $\boldsymbol{\pi}$ is the probability distribution of the initial state, and its entries describe how likely it is to start in a certain state. The full model of an HMM is a tuple referred to as $\lambda = (\mathbf{T}, \mathbf{O}, \boldsymbol{\pi})$, with the following description for each element:

- **Transition Probability Matrix**: $\mathbf{T} \in \mathbb{R}_+^{K \times K}$ with elements $\mathbf{T}_{ij} = P(x_{t+1} = s_i | x_t = s_j)$. Here, $\mathbf{T}_{ij}$ is the probability of transitioning from the current state s_j to the next state s_i, with $\sum_{i=1}^{K} \mathbf{T}_{ij} = 1, \ \forall j$;
- **Emission Probability Matrix**: $\mathbf{O} \in \mathbb{R}_+^{D \times K}$ with elements $\mathbf{O}_{dk} = P(y_t = o_d | x_t = s_k)$. Here, the probability of observing o_d given the current state s_k is given by $\mathbf{O}_{dk}$, with $\sum_{d=1}^{D} \mathbf{O}_{dk} = 1, \ \forall k$;
- **Initial State Probability Distribution**: $\boldsymbol{\pi} \in \mathbb{R}_+^{K}$ with elements $\boldsymbol{\pi}_k = P(s_k | t = t_1)$, where $\boldsymbol{\pi}_k$ is the probability of starting in s_k, with $\sum_{k=1}^{K} \boldsymbol{\pi}_k = 1$.

Figure 2-2 shows what an example of a HMM could represent, by having three weather states and the emissions showing the probability of the air being humid.

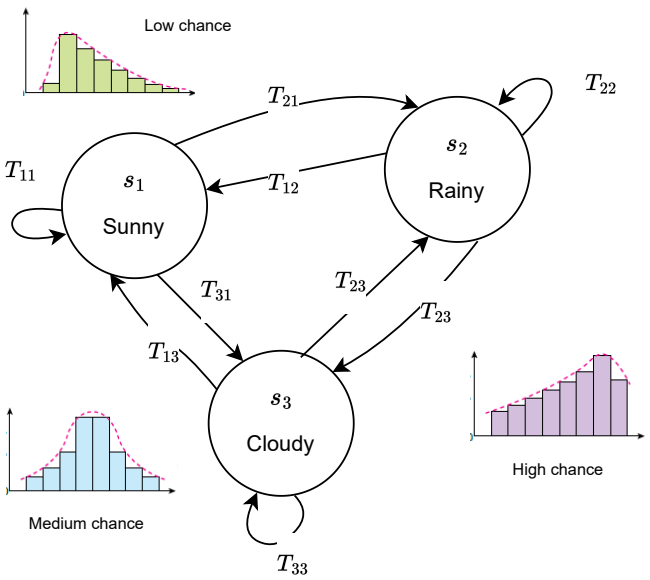


Figure 2-2: An example for a Hidden Markov Model with $K = 3, D = 8$. It shows the probability of transitioning between states and the probability of the observation's value associated with those states. The example here is that the states are Sunny, Rainy, and Cloudy and that the observation is the humidity in the air. When it is Sunny there is a low chance of the air being humid.

Initial distribution and ergodicity of the transition matrix

If the transition matrix $\mathbf{T}$ is ergodic, all states can be reached. The initial distribution $\boldsymbol{\pi}$ can be found by solving $\boldsymbol{\pi}^T\mathbf{T} = \boldsymbol{\pi}^T$, as in Equation (2-1), when the transition matrix is ergodic [11, 12].

$$\boldsymbol{\pi}^T\mathbf{T} = \boldsymbol{\pi}^T \rightarrow \boldsymbol{\pi}^T(\mathbf{T} - \mathbf{I}) = 0. \tag{2-1}$$

With the assumption that the HMM is ergodic, $\boldsymbol{\pi}$ can be found purely based on $\mathbf{T}$. This assumption is made throughout this thesis. The transition matrix for synthetic data is generated with noise, meaning no value in it will be zero, making it ergodic.

2-1-1 Extending HMMs

Multivariate HMMs

A multivariate HMM allows multiple observational values to be emitted per time step from the underlying active state. These different sequences are generally discretized with the same resolution, meaning each time series has an emission matrix with D possible observational values, where D is the resolution of the data. Multivariate HMMs require an extension or redefinition of the emission probability matrix, as now there are N emission matrices $\mathbf{O}^{(n)}$, $\forall n \in [1, N]$:

- **Emission Probability Matrix**: $\mathbf{O}^{(n)} \in \mathbb{R}_+^{D\times K}$ with elements defined in the form $\mathbf{O}_{dk}^{(n)} = P(y_t^{(n)} = o_d^{(n)} | x_t = s_k)$. Here, $\mathbf{O}_{dk}^{(n)}$ is the probability of observation number n at time t having the value of o_d given the current state s_k, with $\sum_{d=1}^{D} \mathbf{O}_{dk}^{(n)} = 1$, $\forall k \in [1, K]$, $n \in [1, N]$.

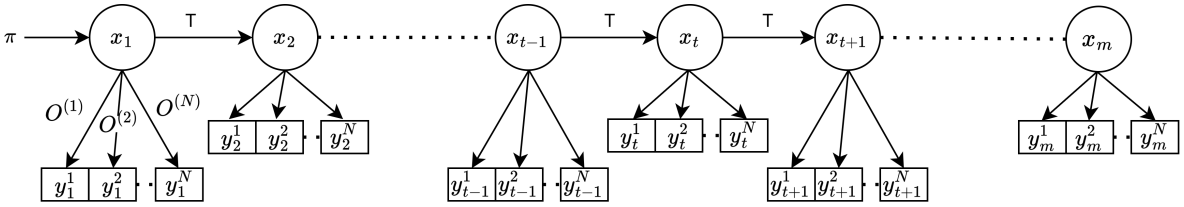


Figure 2-3: A sequence, over time, of multivariate HMM observations and the associated hidden states.

For multivariate HMMs, there exists an emission matrix for each observational sequence. In functional Magnetic Resonance Imaging (fMRI) data, for example, there are ~ 100.000 voxels per time frame or ~ 100.000 observational sequences. As multivariate HMMs quickly become more expensive to learn or fit as the number of observations n increases, it is generally hard to find an HMM for such datasets without performing a preprocessing step such as clustering multiple data sequences together.

Continuous HMM

Discrete HMMs use probability mass functions along the columns of the emission matrix for each state. Continuous HMMs(cHMM) offer a Probability Density Function (PDF) for the emission from the state to the observation. This means that for a univariate HMM, each state has a PDF to describe what the probability of emission is; for a multivariate HMM, each state has n PDFs. In this thesis, unless described otherwise, PDFs are normal distributions/Gaussian distributions. Hence, the following can be defined:

$$\mathbf{O}^{(n)} \in \mathcal{N}^K, \text{ with } \mathbf{O}_k^{(n)} = f(s_k, n) = P(y_t^{(n)} | x_t = s_k) = \mathcal{N}(\mu_k^{(n)}, \sigma_k^{(n)}), \ \forall k \in [1, K], \ n \in [1, N], \tag{2-2}$$

with $\int_{-\infty}^{\infty} f(s_k, n) = 1$, and where $\mathcal{N}$ is a Gaussian distribution.

Higher order HMMs

As seen above, for an HMM we have $\mathbf{T}_{ij} = P(x_{t+1} = s_j | x_t = s_i)$ and $\mathbf{O}_{dk} = P(y_t = o_d | x_t = s_k)$ suggesting that both the next state and the current observation purely depend on the current state. This is a quality of a first-order HMM and is generally described as Markovian behavior or as a Markovian process, and is used in HMMs and Markov chains alike. HMMs can be extended to a second-order HMM, which has states that are all dependent on the previous two states: $\mathbf{T}_{a,b,c} = P(x_{t+1} = s_a | x_t = s_b, x_{t-1} = s_c)$ such that $\mathbf{T} \in \mathbb{R}^{K \times K \times K}$. The idea is that higher-order HMMs can describe more complex probabilistic behavior, at the cost of a more complex fitting process.

Some literature uses the term order of an HMM to describe the number of states that an HMM has; however, in this thesis, order refers to the number of dependencies an event (state or observation) has. Also, in this thesis, only first-order HMMs are considered.

2-1-2 Learning HMMs through Empirical Methods

This section introduces two key algorithms used in the context of HMMs: the Viterbi algorithm and the Baum-Welch algorithm. Although this thesis focuses on tensor-based approaches rather than these traditional methods, they serve as valuable reference points for evaluating performance and correctness.

Viterbi Algorithm

The Viterbi algorithm is used to infer the most likely sequence of hidden states (known as the Viterbi path) that could have generated a given sequence of observations. Rather than estimating the distribution over all possible state sequences, it identifies the single most probable one.

This is done through dynamic programming: at each time step, the algorithm computes the highest probability of any path that ends in a given state, building up these probabilities recursively. Alongside, it keeps track of backpointers to reconstruct the optimal state path once the end of the observation sequence is reached.

The computational complexity of the Viterbi algorithm is linear in the number of time steps m, and quadratic in the number of hidden states K. It is particularly useful when a concrete state sequence is needed, rather than marginal distributions over states.

Baum-Welch Algorithm

The Baum-Welch algorithm is the standard method for learning HMM parameters from data. It is a special case of the Expectation-Maximization (EM) algorithm, and works by iteratively refining estimates of the transition probabilities, emission probabilities, and initial state distribution [13].

Each iteration consists of two steps:

- **Expectation step (E-step)**: Uses the forward-backward algorithm [13] to compute the expected state transitions and state occupancies given the current model parameters.
- **Maximization step (M-step)**: Updates the model parameters to maximize the expected complete data log-likelihood computed in the E-step.

The computational complexity of Baum-Welch per sequence is $\mathcal{O}(K^2 m)$, where K is the number of hidden states and m is the sequence length. This quadratic dependence on K and linear dependence on m means that the algorithm becomes increasingly slow for long sequences or large models. When applied to a dataset with N sequences, the total complexity scales linearly with N, making it computationally expensive for large-scale or high-resolution time series data.

Although widely used and robust, Baum-Welch has limitations. It often converges to local optima, particularly when the number of hidden states is large or the initialization is poor. Nevertheless, its flexibility—including support for multivariate HMMs and continuous emissions—makes it a strong benchmark for comparison.

In this thesis, Baum-Welch serves as a baseline for evaluating the effectiveness of tensor-based methods in univariate and multivariate settings.

2-1-3 HMM summary

This section has shown what HMMs are, how they are defined in this thesis, and how they are generally learned using empirical algorithms. The problem with empirical algorithms, and Baum-Welch in particular, is that it uses a forward-backward pass every iteration, meaning that, as the length of the data increases, the algorithm becomes exponentially slower. Additionally, Baum-Welch tends to get stuck in local minima.

The next section looks at Joint Probability Tensors (JPTs), which aim to view data from a probabilistic perspective and find a different route to the HMM matrices.

2-2 Joint Probability Tensors

This section describes Joint Probability Tensors (JPTs), what they are, how they are found from data, and what probability marginalization can be applied to eventually find the Hidden Markov Model (HMM) matrices from data, as is done in this thesis. Lastly, a few state-of-the-art variants of JPT schemes and their limitations are mentioned.

2-2-1 Description of JPTs

JPTs are collections of probabilities that describe how often combinations of data appear. Say there is an observed data sequence $y = [y_1, \ldots, y_m]$, with length m, that is sampled from the alphabet $\mathcal{Y} = [o_1, \ldots, o_D]$, where $D \ll m$. Then, for singletons, pairs, and triplets of data of y, the following can be defined [5, 14, 15, 16]:

$$\begin{aligned} \mathbf{p}_d &= P(y_t = o_d), \\ \mathbf{P}_{d_1 d_2} &= P(y_t = o_{d_1}, y_{t+1} = o_{d_2}), \\ \underline{\mathbf{P}}_{d_1 d_2 d_3} &= P(y_{t-1} = o_{d_1}, y_t = o_{d_2}, y_{t+1} = o_{d_3}), \end{aligned} \tag{2-3}$$

where $\mathbf{P} \in \mathbb{R}_+^D$ with $\sum_d^D vec(\mathbf{P})_d = 1$, $\mathbf{P} \in \mathbb{R}_+^{D \times D}$ with $\sum_d^{D^2} vec(\mathbf{P})_d = 1$, and $\underline{\mathbf{P}} \in \mathbb{R}_+^{D \times D \times D}$ with $\sum_d^{D^3} vec(\underline{\mathbf{P}})_d = 1$. This idea can be extended to any sub-collection with window length w of the data Y:

$$\underline{\mathbf{P}}_{d_1, \ldots, d_W} = P(y_t = o_{d_1}, \ldots, y_{t+W-1} = o_{d_W}), \tag{2-4}$$

where $d_i \in [1, \ldots, D] \ \forall i$ and $\underline{\mathbf{P}} \in \mathbb{R}_+^{[D]^W}$ with $\sum_d^{D^W} vec(\underline{\mathbf{P}})_d = 1$, and $[D]^W = \underbrace{D \times \cdots \times D}_{W \text{ times}}$. This ensures that the sum of all entries of the JPT adds up to 1.

2-2-2 Finding JPTs from data

These JPTs can be approximated from data through the indicator function [9]. The indicator function returns 1 if the value inserted belongs to a set, and 0 if that value does not belong to the set—see Equation (2-5). This set is the finite, discrete alphabet of the observations used in HMMs (Section 2-1). Note that, when working with real data that does not come in a discrete finite set or has a resolution that is deemed too large, the data needs to be binned into D bins before being fed to the indicator function.

$$\begin{aligned} &\mathbf{I}_{\mathcal{A}} : X \to 0, 1 \\ &\quad \text{defined as:} \\ &\mathbf{I}_{\mathcal{A}}(x) := \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases} \end{aligned} \tag{2-5}$$

The indicator function can also be used for vectors, but it must be noted that then, each element in said vector has its own set $\mathcal{A}$ (which is usually just a single value) or is associated

in order with a value from a common $\mathcal{A}$. The indicator function is then used as follows to obtain the JPT:

$$ {}^{m}\underline{\mathbf{P}}_{d_1,\dots,d_w} = \alpha \sum_{t=1}^{m-W+1} \mathbf{I}_{\mathcal{A}}(y_t,\dots,y_{t+W-1}), \tag{2-6} $$

where $\mathcal{A} = \mathcal{Y}$ is the finite discrete alphabet of observations, and α is a scaling coefficient chosen such that $\sum_{d_1,\dots,d_w} \underline{\mathbf{P}}_{d_1,\dots,d_w} = 1$. As the total sum over all entries of $\underline{\mathbf{P}}$ needs to equal one, and every one of the $m-W+1$ triplets in the m length data sequence adds a 'one' value to the tensor [9], the scaling parameter α therefore takes the value of $1/(m-W+1)$.

In this thesis, a window size and, therefore, tensor dimension size of 3 are used, leading to the following specification of Equation (2-6):

$$ \underline{\mathbf{P}}_{d_a,d_b,d_c} = \frac{1}{m-2} \sum_{t=2}^{m-1} \mathbf{I}_{\mathcal{A}}(y_{t-1},y_t,y_{t+1}). \tag{2-7} $$

In practice, this equation is much simpler to implement by simply looping over all $m-2$ triplets, and since they can be pre-processed and discretized into indices, adding a one to that entry, then finalizing by dividing every entry in the obtained tensor by $m-2$. An example of the result of this is given in Figure 2-4.

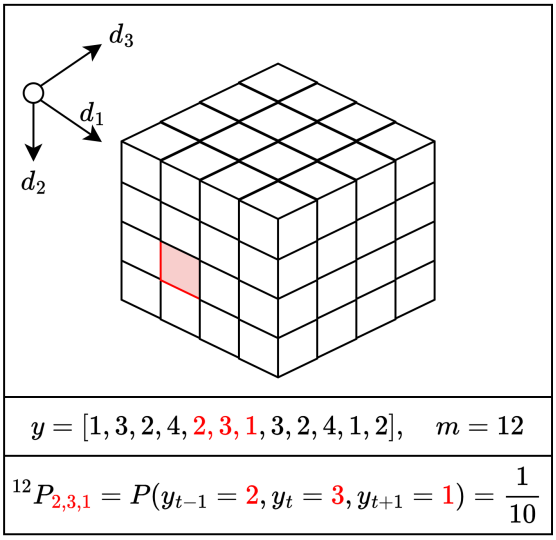


Figure 2-4: An example of a JPT sampled from a very short data sequence, for which $D = 4, m = 12$, having 10 consecutive triplets, making the red entry ${}^{12}P_{2,3,1} = 1/10$. This also showcases the importance of having enough samples.

2-2-3 Underlying structure in JPTs through decomposition

This thesis will focus on three-way JPTs. The three-way tensor $\underline{\mathbf{P}} = P(y_{t-1}, y_t, y_{t+1})$ represents the joint probability of three consecutive values of data. With regards to HMMs, it is known that these three data samples are generated from three consecutive states $\{x_{t-1}, x_t, x_{t+1}\}$.

Using this knowledge, this JPT can be described by the following probability marginalization [7, 17]:

$$\begin{aligned}\underline{\mathbf{P}}_{d_a,d_b,d_c} &= P(y_{t-1}=o_{d_a}, y_t=o_{d_b}, y_{t+1}=o_{d_c}) \\ &= \sum_{k=1}^{K} \underbrace{P(y_{t-1}=o_{d_a}|x_t=s_k)}_{\mathbf{A}_{d_a,k}} \underbrace{P(y_t=o_{d_b}|x_t=s_k)}_{\mathbf{B}_{d_b,k}=\mathbf{O}_{d_b,k}} \underbrace{P(y_{t+1}=o_{d_c}|x_t=s_k)}_{\mathbf{C}_{d_c,k}} \underbrace{P(x_t=s_k)}_{\lambda_k}.\end{aligned} \tag{2-8}$$

These marginal probabilities concerning the three different states are conditioned into matrices $\mathbf{A} = P(y_{t-1}|x_t) \in \mathbb{R}^{D\times K}, \mathbf{B} = P(y_t|x_t) \in \mathbb{R}^{D\times K}, \mathbf{C} = P(y_{t+1}|x_t) \in \mathbb{R}^{D\times K}$. These conditional probabilities can be described using the HMM matrices [15, 14]:

$$\mathbf{A} = \mathbf{O}\text{diag}(\boldsymbol{\pi})\mathbf{T}^T\text{diag}(\mathbf{T}\boldsymbol{\pi})^{-1}, \quad \mathbf{B} = \mathbf{O}, \quad \mathbf{C} = \mathbf{O}\mathbf{T}. \tag{2-9}$$

To obtain the factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$, a tensor decomposition has to be applied to the JPT $\underline{\mathbf{P}}$. The next section, Section 2-3, will discuss a tensor decomposition method, i.e., Canonical Polyadic Decomposition (CPD), how it works, and how it can be used to obtain the aforementioned factor matrices from a JPT.

Related work: alternative JPT decomposition structures

This subsection suggests alternative methods to decompose JPTs to obtain HMMs, which are commonly used in the literature, but will not be further built upon in this thesis.

One way to extract information from a JPT is by decomposing it in relation to HMMs [9, 6, 8, 18] in the following tensor train format:

$$\underline{\mathbf{P}}_{d_1,\ldots,d_w} = \boldsymbol{\pi}^T\mathbf{A}^T(d_1)\ldots\mathbf{A}^T(d_w)\mathbf{1}_K, \tag{2-10}$$

where $\mathbf{1}_K$ is a column vector with length K filled with ones, and $\mathbf{A}(d) = \mathbf{T}\text{diag}(\mathbf{O}_{d,1}, \ldots, \mathbf{O}_{d,K})$, with $d = [1, \ldots, D]$, K equal to the number of states, and D the number of observational bins. This decomposition has the benefit that it can create tensors of any mode matching the window size used in the indicator function. This makes it so that the information that can be captured goes beyond triplets of data, such as quadruplets or quintuplets. This decomposition structure lacks the convenience of obtaining meaningful matrices, as the $\mathbf{A}(d)$ matrices require an extra decomposition step to obtain the HMM matrices, adding an extra layer of error. Multiple decomposition steps of matrices/tensors also do not result in unique decompositions, as many combinations of different decomposed matrices can result in the same tensor. This non-uniqueness is also a property of Tensor-Train decompositions. Because of a twofold non-uniqueness, the space in which possible HMM matrices live increases significantly, whilst a CPD of the JPT in the previous section can be unique.

Another type of JPT, $\underline{\mathbf{L}} \in \mathbb{R}^{K\times K\times D}$, can also be directly decomposed [6] into HMM matrices $\underline{\mathbf{L}} = [[\mathbf{T}, \mathbf{I}_K, \mathbf{O}]]$. The decomposed tensor is structured as follows:

$$\begin{aligned}\underline{\mathbf{L}}_{ikd} &= P(x_{t+1}=i|x_t=k)\cdot P(y_t=d|x_t=k) \\ &= P(x_{t+1}=i, y_t=d|x_t=k).\end{aligned}$$

The problem with this method is that this tensor is not obtainable given a sequence of data. The tensor is created using $\mathbf{T}_{ik} = P(x_{t+1} = i | x_t = k)$ and $\mathbf{O}_{dk} = P(y_t = d | x_t = k)$, which are the matrices that HMM learning aims to uncover.

A fourth method [18] constructs a third order tensor $\underline{\mathbf{M}}$ of the shape $\underline{\mathbf{M}} \in \mathbb{R}^{D^w \times D^w \times D}$, where w is defined by the window-size $W = 2w + 1$, and D is the number of emissions. The tensor is then defined as:

$$\underline{\mathbf{M}}_{L(l_1^w), L(l_{-1}^{-w}), l_0} = P(y_{-w}^w = l_{-w]}^w), \forall l_{-w}^w \in [D]^W.$$

This uses a bijective mapping $L : [D]^w \to [D^w]$ of the multi-index $l_1^w = (l_1, \ldots, l_w) \in [D]^w$ to the scalar index $L(l_1^w) = (l_1 - 1)D^{w-1} + (l_2 - 1)D^{w-2} + \cdots + l_w \in [D^w]$. This tensor $\underline{\mathbf{M}}$ can then be decomposed into factor matrices $\underline{\mathbf{M}} = \mathbf{A} \times_1 \mathbf{B} \times_2 \mathbf{C}$, where $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{D^w \times K}$ where K is the number of states, and $\mathbf{C} \in \mathbb{R}^{D \times K}$. The HMM matrices can then be recovered through:

$$\begin{aligned}
\mathbf{O}_{:,k} &= \mathbf{C}_{:,k} / (1^T \mathbf{C}_{:,k}), \quad \forall k \in [1, \ldots, K] \\
\mathbf{T} &= \left(\mathbf{O} \odot \mathbf{A}^{(n-1)} \right)^\dagger \mathbf{A}, \quad \text{if } \mathbf{A} \text{ has full column rank } K \\
\mathbf{T} &= \mathbf{O}^\dagger \mathbf{A}^{(1)}, \quad \text{if } \mathbf{C} \text{ has full column rank } K
\end{aligned}$$

where $\mathbf{A}^{(1)} \in \mathbb{R}^{D \times K}$ and is the first D rows of $\mathbf{A}$, and $\mathbf{A}^{(n-1)} \in \mathbb{R}^{D^{w-1} \times K}$ is the rest of the rows of $\mathbf{A}$.

A notable problem with this method is the large dimension sizes of the tensor and its factor matrices with an increase in the number of emissions D and especially in the variable w that has an effect on the window size W. This quickly leads to memory issues for both the storage of the tensor and a CPD to decompose it.

Finally, in [8], tensor decompositions are used to obtain higher-order HMMs. This causes the transition matrix to become $\underline{\mathbf{T}} \in \mathbb{R}^{K \times \cdots \times K}$.

Spectral methods for JPTs

Another method to find HMM matrices from JPTs without tensor decomposition is spectral algorithms [5, 14], and will also not be further used in this thesis. Spectral algorithms look at the singular values and vectors. A parallel state is used that is a linear transformation of the actual HMM states [5], and is named the observable representation. It requires the definition of a matrix U that consists of the left singular vectors of $[\mathbf{P}_{2,1}]_{ij} = P(y_t = o_i, y_{t-1} = o_j)$ which correspond to non-zero singular values. Also needed are $[\mathbf{p}_1]_i = P(y_t = o_i)$ and $[\underline{\mathbf{P}}_{3,x,1}]_{ij} = P(y_{t+1} = o_i, y_t = o_x, y_{t-1} = o_j)$, which then leads to the observable representation $\forall x \in [d]$:

$$\begin{aligned}
\mathbf{b}_1 &= \mathbf{U}^T \mathbf{p}_1, \\
\mathbf{b}_\infty &= (\mathbf{P}_{2,1} U)^\dagger \mathbf{p}_1, \\
\mathbf{B}_x &= (\mathbf{U}^T \underline{\mathbf{P}}_{3,x,1})(U^T \mathbf{P}_{2,1})^\dagger.
\end{aligned}$$

These states are not probability distributions over hidden states, like in an HMM. The actual

HMM matrices can then be recovered using:

$$\mathbf{O} = \begin{bmatrix} \lambda_{1,1} & \lambda_{1,2} & \cdots & \lambda_{1,k} \\ \lambda_{2,1} & \lambda_{2,2} & \cdots & \lambda_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{d,1} & \lambda_{d,2} & \cdots & \lambda_{d,k} \end{bmatrix}$$

where $\lambda_{r,1}, \ldots, \lambda_{r,k}$ are the eigenvalues of $(\mathbf{U}^T \underline{\mathbf{P}}_{3,r,1})(\mathbf{U}^T \mathbf{P}_{3,1})^\dagger, \ \forall r \in [1, d]$. Then, $\boldsymbol{\pi} = \mathbf{O}^\dagger \mathbf{p}_1$ and $\mathbf{T} = \mathbf{O}^\dagger \mathbf{P}_{2,1} (\mathbf{O}^\dagger)^T \text{diag}(\boldsymbol{\pi})^{-1}$.

A very similar spectral method is used in [14], and both are implemented in [16], which shows practical learning of HMMs with $k = 2, 3$ and $d = 3, 6, 8, 10$. However, this implementation shows that for $k = 3$ the algorithm does not converge, and both spectral algorithms scale badly with an increase in K and D.

2-2-4 JPT summary

JPTs are a collection of the probability of observing multiple consecutive values in data. In this thesis, consecutive triplets are used to create a three-way JPT $\underline{\mathbf{P}}$, which is then decomposed into factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$. These, in turn, can be used to find the HMM matrices $\mathbf{T}$ and $\mathbf{O}$ as observed in Equation (2-8) and Equation (2-9). The method for decomposing the JPT into these factor matrices is discussed in Section 2-3.

2-3 Canonical Polyadic Decomposition

The Canonical Polyadic Decomposition (CPD) of a tensor $\underline{\mathbf{X}}$ is its minimal decomposition into a sum of rank-1 tensors [19]:

$$\underline{\mathbf{X}} = \underline{\boldsymbol{\Lambda}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \cdots \times_N \mathbf{B}^{(N)} = \sum_{r=1}^{R} \lambda_r \mathbf{b}_r^{(1)} \circ \mathbf{b}_r^{(2)} \cdots \circ \mathbf{b}_r^{(N)}. \tag{2-11}$$

A CPD can be denoted as:

$$\underline{\mathbf{X}} = [[\underline{\boldsymbol{\Lambda}}; \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(N)}]] \quad \text{or} \quad [[\mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(N)}]], \tag{2-12}$$

where $\mathbf{B}^{(n)}$ are called factor matrices, and $\mathbf{b}_r^{(n)}$ are the signatures. CPD is also often referred to as PARAFAC or CANDECOMP.

A CPD, $\underline{\mathbf{X}} = [[\mathbf{A}, \mathbf{B}, \mathbf{C}]]$, is unique if $k_A + k_B + k_C \geq 2R + 2$, for $R = rank(\underline{\mathbf{X}})$, where k_A is the Kruskal-rank of A—the maximum value of k such that any k columns of $\mathbf{A}$ are linearly independent.

An N-way tensor CPD, $\underline{\mathbf{X}} = [[\mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(N)}]]$, is unique if the following holds:

$$\sum_{n=1}^{N} k_{B^{(n)}} \geq 2R + (N - 1). \tag{2-13}$$

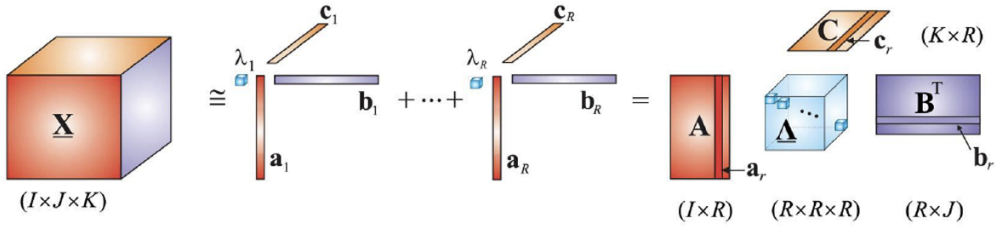


Figure 2-5: A graphical representation of a three-way CPD [20, 21, 22, 23].

If the factor matrices are not yet computed, it can estimate whether the CPD is expected to be unique. The tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$ of rank R has a unique rank-R CPD if:

$$R \leq K \quad \text{and} \quad R(R-1) \leq I(I-1)J(J-1)/2. \tag{2-14}$$

In real cases, a CPD needs to be fit, as there is noise and other artifacts in the data. The CPD model then looks as follows:

$$\underline{\mathbf{X}} = \underline{\mathbf{\Lambda}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \cdots \times_N \mathbf{B}^{(N)} + \underline{\mathbf{E}} = \sum_{r=1}^{R} \lambda_r \mathbf{b}_r^{(1)} \circ \mathbf{b}_r^{(2)} \cdots \circ \mathbf{b}_r^{(N)} + \underline{\mathbf{E}}. \tag{2-15}$$

The fitting of this CPD is done my minimizing a suitable cost function that can be defined as $J = \min ||\underline{\mathbf{X}} - [[\mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(N)}]]||_F^2$. This can be achieved with many optimization algorithms, but alternating least squares (ALS) is often used as it uses linear least squares at each iteration, and is, therefore, fast.

Finally, one can note that the structure of Equation (2-11) for the tensor and the factor matrices or vectors is identical to the structure of the probability of Equation (2-8). This is precisely why CPD can be used to decompose the JPT into factor matrices that have a probabilistic meaning—see Equation (2-16). Here, the rank of the JPT, R, is equal to the order of the HMM, K.

$$\underline{\mathbf{P}} = \underline{\mathbf{\Lambda}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} = \sum_{k=1}^{K} \lambda_k \mathbf{A}_{:,k} \circ \mathbf{B}_{:,k} \circ \mathbf{C}_{:,k} \tag{2-16}$$

Note that the factor matrices have a probabilistic meaning $\mathbf{A} = P(y_{t-1}|x_t)$, $\mathbf{B} = P(y_t|x_t)$, $\mathbf{C} = P(y_{t+1}|x_t)$, so their columns need to be stochastic (positive values and sum up to one), resulting in the scaling vector λ being obsolete, which is why in this thesis it is often omitted.

2-4 Summary

This preliminary chapter discussed Hidden Markov Models (HMMs), including extensions to multivariate and continuous settings. The chapter then discussed Joint Probability Tensors (JPTs), which are generated through data sequences and describe the probability of observing multiple consecutive values in data. The JPTs can then be decomposed into factor matrices using Canonical Polyadic Decomposition (CPD), which is also explained in this chapter. In Chapter 3, these preliminaries will be used to complete the multivariate HMM learning method using CPD, which is the core of this thesis.

Chapter 3

Methods for Learning Hidden Markov Models

This chapter presents the methods used in this report for learning Hidden Markov Models (HMMs). It starts by explaining how parts from the previous chapters are used together to learn univariate HMMs through uncoupled Canonical Polyadic Decomposition (CPD), in Section 3-1. Then, Section 3-2 speaks on how to couple many univariate problems together to create a multivariate problem by using a common transition matrix $\mathbf{T}$ between all these problems. Finally, Section 3-3 presents a few considerations to have with these methods, measures that can be used to define performance, and functions that can be used to describe the behavior of the methods.

3-1 Univariate HMM learning through uncoupled CPD

To obtain a Hidden Markov Model (HMM) from a univariate data sequence through Canonical Polyadic Decomposition (CPD), as seen in Chapter 2, three steps (Figure 3-1) are taken:

1. The observational sequence y (time series) is turned into a Joint Probability Tensor (JPT) $\underline{\mathbf{P}}$ using the indicator function.
2. The JPT $\underline{\mathbf{P}}$ is decomposed into factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$.
3. The factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are turned into the HMM matrices $\mathbf{T}$ and $\mathbf{O}$.

Here follows a small reiteration of these steps mathematically, as in the next section, these ideas will be expanded upon. As explained in Section 2-2, the JPT $\underline{\mathbf{P}}$ can be obtained from a data sequence y using Equation (2-7) [9]:

$$ {}^{m}\underline{\mathbf{P}}_{d_a,d_b,d_c} \approx \alpha \sum_{t=1}^{m-2} \mathbf{I}_{\mathcal{A}}(y_{t-1} = o_{d_a}, y_t = o_{d_b}, y_{t+1} = o_{d_c}). $$

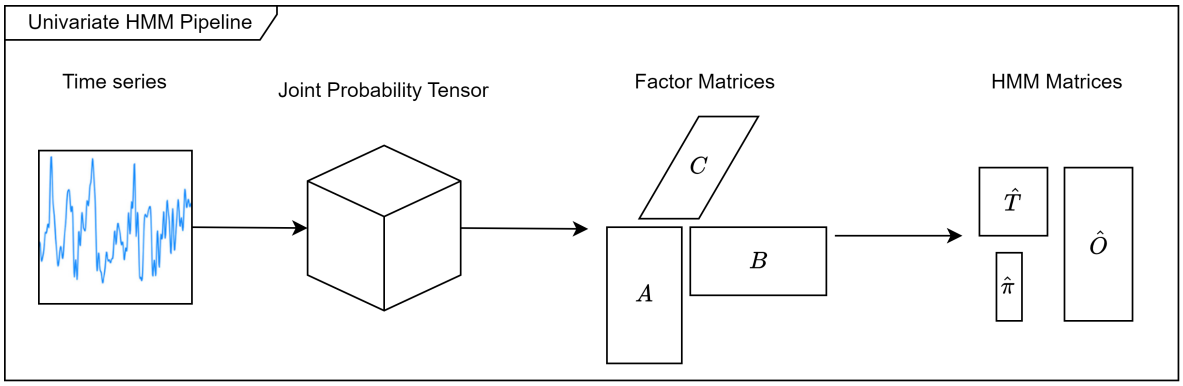


Figure 3-1: A diagram showing the pipeline for applying univariate HMM learning through CPD to obtain HMM matrices.

Now that ${}^{m}\underline{\mathbf{P}}$ can be obtained from a data sequence with length m, the tensor can be decomposed using CPD. The tensor $\underline{\mathbf{P}}$ represents $P(y_{t-1}, y_t, y_{t+1})$. Hence, as seen in Section 2-3, using Equation (2-8) and Equation (2-11), the decomposition in a probability setting can be broken down as follows [7, 17]:

$$\begin{aligned}
\underline{\mathbf{P}}_{d_a,d_b,d_c} &= P(y_{t-1} = o_{d_a}, y_t = o_{d_b}, y_{t+1} = o_{d_c}) \\
&= \sum_{k=1}^{K} \underbrace{P(y_{t-1} = o_{d_a} | x_t = s_k)}_{\mathbf{A}_{d_a,k}} \underbrace{P(y_t = o_{d_b} | x_t = s_k)}_{\mathbf{B}_{d_b,k} = \mathbf{O}_{d_b,k}} \underbrace{P(y_{t+1} = o_{d_c} | x_t = s_k)}_{\mathbf{C}_{d_c,k}} \underbrace{P(x_t = s_k)}_{\lambda_k}, \\
\underline{\mathbf{P}} &= \sum_{k=1}^{K} \lambda_k \mathbf{A}_{:,k} \circ \mathbf{B}_{:,k} \circ \mathbf{C}_{:,k}.
\end{aligned}$$

It can be noted that when the individual probabilities become conditioned into matrices the following representations emerge: $\mathbf{A} = P(y_{t-1}|x_t)$, $\mathbf{B} = P(y_t|x_t)$, $\mathbf{C} = P(y_{t+1}|x_t)$. According to [15, 14], these can then be put into terms of the HMM matrices:

$$\begin{aligned}
\mathbf{A} &= \mathbf{O}\text{diag}(\boldsymbol{\pi})\mathbf{T}^T\text{diag}(\mathbf{T}\boldsymbol{\pi})^{-1}, \\
\mathbf{B} &= \mathbf{O}, \\
\mathbf{C} &= \mathbf{O}\mathbf{T}.
\end{aligned}$$

Note that this means that the tensor $\underline{\mathbf{P}}$ can also be put in terms of these $\mathbf{A}, \mathbf{B}, \mathbf{C}$ matrices using the notation from Equation (2-12), which allows the following equation to arise:

$$\underline{\mathbf{P}} = [[\mathbf{A}, \mathbf{B}, \mathbf{C}]] = [[\mathbf{O}\text{diag}(\boldsymbol{\pi})\mathbf{T}^T\text{diag}(\mathbf{T}\boldsymbol{\pi})^{-1}, \mathbf{O}, \mathbf{O}\mathbf{T}]]. \tag{3-1}$$

This can then be used to obtain an estimate of the HMM matrices $\hat{\mathbf{O}} = \mathbf{B}$ and $\hat{\mathbf{T}} = \mathbf{B}^{\dagger}\mathbf{C}$. As noted in Section 2-1, $\hat{\boldsymbol{\pi}}$ can be obtained from $\hat{\mathbf{T}}$, given that the HMM is ergodic. To uncover the factor matrices given only the JPT, the following minimization problem can be defined:

$$\begin{aligned}
\mathbf{A}, \mathbf{B}, \mathbf{C} = \underset{\mathbf{A},\mathbf{B},\mathbf{C}}{\arg\min} \; & ||\underline{\mathbf{P}} - [[\mathbf{A}, \mathbf{B}, \mathbf{C}]]||_F^2 \\
s.t. \quad & \\
& \mathbf{A}, \mathbf{B}, \mathbf{C} > 0 \;\; \text{(element wise)} \\
& \textstyle\sum \mathbf{A}_{:,k} = 1, \sum \mathbf{B}_{:,k} = 1, \sum \mathbf{C}_{:,k} = 1, \;\; \forall i \in [1, .., K]
\end{aligned} \tag{3-2}$$

3-1-1 Alternating Least Squares

A common workhorse for such a minimization scheme, and CPD in general, is Alternating Least Squares (ALS). ALS generates separate minimization problems for all factor matrices by fixing all other factor matrices. Meaning for the problem of finding the local optimum for $\hat{\mathbf{A}}$ the factor matrices $\mathbf{B}$ and $\mathbf{C}$ get fixed, after which $\hat{\mathbf{A}}$ gets calculated using a least squares approach [20, 21]. This gives rise to the following minimization subproblems:

$$\begin{aligned}\hat{\mathbf{A}} &\leftarrow \arg\min_{\hat{\mathbf{A}}} ||\mathbf{P}_{(1)} - \hat{\mathbf{A}}(\mathbf{C} \odot \mathbf{B})^T||_F^2, \\ \hat{\mathbf{B}} &\leftarrow \arg\min_{\hat{\mathbf{B}}} ||\mathbf{P}_{(2)} - \hat{\mathbf{B}}(\mathbf{C} \odot \mathbf{A})^T||_F^2, \\ \hat{\mathbf{C}} &\leftarrow \arg\min_{\hat{\mathbf{C}}} ||\mathbf{P}_{(3)} - \hat{\mathbf{C}}(\mathbf{B} \odot \mathbf{A})^T||_F^2,\end{aligned} \tag{3-3}$$

where $\mathbf{P}_{(i)}$ is the i-th unfolding of the tensor $\underline{\mathbf{P}}$. By iteratively solving these minimization subproblems and thus alternating between them, ALS aims to minimize the problem as a whole. These subproblems can be written using a closed-form expression [20, 21]:

$$\begin{aligned}\hat{\mathbf{A}} &= \mathbf{P}_{(1)}(\mathbf{C} \odot \mathbf{B}) \left(\mathbf{C}^T\mathbf{C} \circledast \mathbf{B}^T\mathbf{B}\right)^\dagger, \\ \hat{\mathbf{B}} &= \mathbf{P}_{(2)}(\mathbf{C} \odot \mathbf{A}) \left(\mathbf{C}^T\mathbf{C} \circledast \mathbf{A}^T\mathbf{A}\right)^\dagger, \\ \hat{\mathbf{C}} &= \mathbf{P}_{(3)}(\mathbf{B} \odot \mathbf{A}) \left(\mathbf{B}^T\mathbf{B} \circledast \mathbf{A}^T\mathbf{A}\right)^\dagger,\end{aligned} \tag{3-4}$$

where $\odot$ is the Khatri-Rao product, and $\circledast$ is the Hadamard product (element-wise multiplication). The full derivation of these update equations can be found in Appendix B-1. Note that after each update step for the three factor matrices, they are made stochastic using Equation (3-14), which is discussed later in this chapter. This also completes the constraints defined in the minimization problem above Equation (3-2).

Through this optimization scheme, the factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are uncovered, which can then be used to obtain the HMM matrices. As seen in Equation (2-9), $\mathbf{B} = \hat{\mathbf{O}}$ and $\mathbf{C} = \hat{\mathbf{O}}\hat{\mathbf{T}}$. This means the emission matrix is found in $\mathbf{B}$, and the transition matrix can be found by $\hat{\mathbf{T}} = \mathbf{B}^\dagger\mathbf{C}$.

3-1-2 Sensitivity Issues

CPD aims to obtain factor matrices that give a good reconstruction of the original tensor. CPD itself does not aim to find good HMM matrices. Because of this, errors or biases in the JPT give bad estimates for the HMM matrices even when the reconstruction error between the factor matrices and JPT is small. There are a couple of factors that influence the ability of CPD to find good estimates of the HMM matrices. A limited number of samples, causing an incomplete description of the underlying distribution in the JPT. Noise in the measurements or observations that flattens out the distributions, making them less characteristic, and effectively increasing the variance of the underlying distribution in the estimate. High correlation of columns in the factor matrices or underlying emission matrix, which makes it hard to distinguish between the different emissions, and thus different states. All of these effects will be discussed separately in the following paragraphs.

Number of samples

As mentioned, the true JPT can be constructed through the factor matrices, and thus HMM matrices as $\underline{\mathbf{P}} = [[\mathbf{O}\text{diag}(\boldsymbol{\pi})\mathbf{T}^T\text{diag}(\mathbf{T}\boldsymbol{\pi})^{-1}, \mathbf{O}, \mathbf{OT}]]$. It can also be approximated using a data sequence with length m. A question that then remains is how closely does the approximated tensor need to be to the true tensor, the calculated one, to retrieve the HMM matrices? Or what influence does the sample size used to construct the JPT have over the recovery of the HMM matrices? To reiterate, the formula for sampling the JPT is defined as:

$$ {}^{m}\underline{\mathbf{P}}_{d_a,d_b,d_c} = \lim_{m\to\infty} \alpha \sum_{t=1}^{m-2} \mathbf{I}_{\mathcal{A}}(y_{t-1} = o_{d_a}, y_t = o_{d_b}, y_{t+1} = o_{d_c}). $$

As can be observed in the above equation, the accuracy of the JPT is dependent on the number of samples m. One can imagine that if ${}^{\mathbf{m}}\underline{\mathbf{P}} \in \mathbb{R}^{D\times D\times D}$ where $D = 10$ and $m = 100$, there are not enough samples, as ${}^{\mathbf{m}}\underline{\mathbf{P}}$ has 1000 entries but is approximated using only 100 samples. This also shows that as the number of samples increases, the approximated JPT ${}^{\mathbf{m}}\underline{\mathbf{P}}$ converges to the true JPT ${}^{\infty}\underline{\mathbf{P}}$. One can also note that as D increases, the convergence slows down, meaning that for a larger D, there is an increasing need for more samples. In

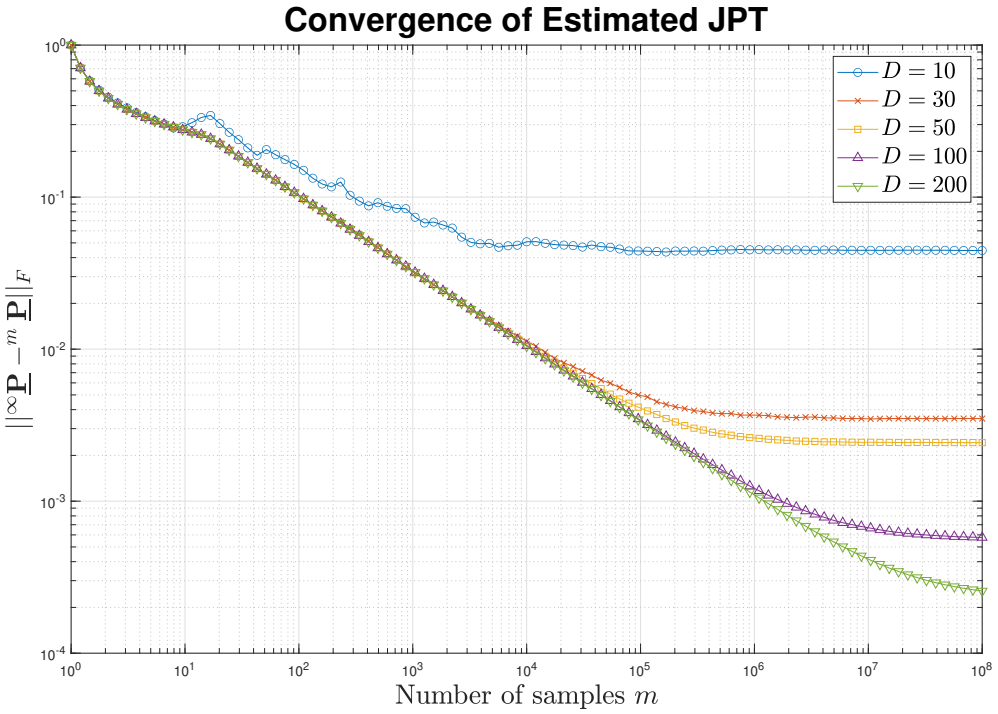


Figure 3-2: A plot of the convergence of ${}^{m}\underline{\mathbf{P}}$ to ${}^{\infty}\underline{\mathbf{P}}$ over the number of samples. The plot also shows different curves for a different number of possible observation values D.

Figure 3-2, one can observe that the error between the approximated JPT and the true JPT converges to a constant value instead of exactly zero. This means that even though more samples are introduced, at some point, the approximated JPT does not converge further to the true underlying JPT.

Noise in measurements

Real data is noisy, which has effects on the ability to estimate the underlying distribution. When estimating a distribution, noise causes the estimate to be flatter than the original distribution. The distribution of the data becomes:

$$X \in \underbrace{\mathcal{N}_S(\mu_S, \sigma_S^2)}_{\text{True distribution}}, \quad Y \in \underbrace{\mathcal{N}_N(0, \sigma_N^2)}_{\text{Noise}}, \quad \text{then} \quad X + Y \in \mathcal{N}(\mu_S, \sigma_S^2 + \sigma_N^2).$$

An example in terms of distributions of the flattening effect that additive noise has on the signal that results in data can be observed in Figure 3-3. This flattening effect also shows

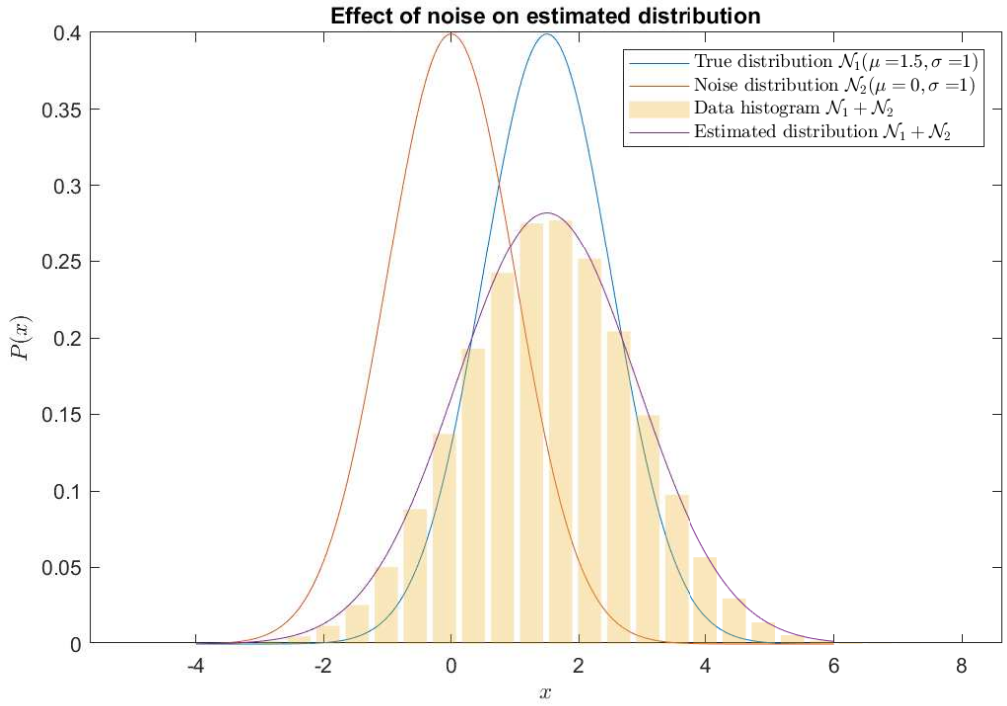


Figure 3-3: A show of the effect of noise on the capability to estimate the underlying distribution.

up when working with the JPT. As this is also a distribution $P(y_{t-1}, y_t, y_{t+1})$, the JPT gets more uniform the higher the SNR. This in turn causes the distributions of the emission matrix $P(y_t|x_t)$ to be flatter also. Flatter distributions along the emission matrix columns mean fewer distinct features, leading to more errors in the CPD process. The effect of this flattening on the signal-to-noise ratio (SNR) is further discussed in Section 3-3-5.

Correlation between emission matrix columns

As mentioned before, both from a CPD and HMM perspective, it is imperative to have characteristic features in the distributions associated with each state. That is to say, if there is a large correlation between two columns of the emission matrix, meaning a large similarity

in distribution, it becomes hard to uncover. This will often result in more failures to find one of the underlying distributions of the emission matrix. This means there is also a limitation as to what HMM matrices can be learned, and thus what the ground truth can be initialized as when working with synthetic data. The full extent of the effect of correlations in the emission matrix will be further discussed in Section 3-3-2, and its effects will be shown in Section 4-1-1.

3-1-3 Solution

Up until now, only univariate HMMs and the accompanying decomposition have been discussed. Meaning, only a single data sequence is considered. As seen in the last couple of sections, this process is sensitive to biases in the JPT, noise, and high correlations in the emission matrix. This means that if multivariate HMMs are considered in the current framework, some of the parallel processes will fail. These failures will result in different transition matrices $\mathbf{T}^{(n)}$ for all of the parallel univariate solutions to the multivariate problem. As these have different permutations, since CPD suffers from permutation ambiguity and different values due to sensitivities, there is no way to permutation match these parallel processes, and the errors cannot be averaged out. In multivariate HMMs, the transition matrix $\mathbf{T}$ is common among all N data sequences, but cannot be found by solving N univariate CPD learning problems. To be able to extend this framework to a multivariate setting, a common transition matrix $\mathbf{T}$ needs to be inferred from all N problems.

3-2 Multivariate HMM learning through Coupled CPD

This section brings the novel method of coupling all separate Hidden Markov Model (HMM) learning problems together via a common transition matrix $\mathbf{T}$ and initial distribution π. To extend Canonical Polyadic Decomposition (CPD) learning to a multivariate setting for HMMs, Equation (3-1) needs to be extended for all N data sequences:

$$\underline{\mathbf{P}}^{(n)} = [[\mathbf{A}^{(n)}, \mathbf{B}^{(n)}, \mathbf{C}^{(n)}]] = [[\mathbf{O}^{(n)}\text{diag}(\boldsymbol{\pi})\mathbf{T}^T\text{diag}(\mathbf{T}\boldsymbol{\pi})^{-1}, \mathbf{O}^{(n)}, \mathbf{O}^{(n)}\mathbf{T}]], \quad \forall n \in [1, \ldots, N]. \tag{3-5}$$

Here it can be observed that both the transition matrix $\mathbf{T}$ and the initial distribution $\boldsymbol{\pi}$ are the same for all N data sequences. This results in the following uncoupled optimization subproblems:

$$J^{(n)} = ||\underline{\mathbf{P}}^{(n)} - [[\mathbf{A}^{(n)}, \mathbf{B}^{(n)}, \mathbf{C}^{(n)}]]||_F^2 \quad \forall n \in [1, \ldots, N]. \tag{3-6}$$

To constrain this common transition matrix $\mathbf{T}$ and initial distribution $\boldsymbol{\pi}$ in all subproblems, the Alternating Least Squares (ALS) update rules found in Equation (3-4) can be extended using Equation (3-5), to obtain two update steps per factor matrix $\mathbf{A}^{(n)}, \mathbf{B}^{(n)}, \mathbf{C}^{(n)} \quad \forall n \in [1, \ldots, N]$, as follows:

$$
\begin{aligned}
&\mathbf{A}^{(n)} \leftarrow \mathbf{P}_{(1)}(\mathbf{C}^{(n)} \odot \mathbf{B}^{(n)}) \left[\left((\mathbf{C}^{(n)})^T\mathbf{C}^{(n)}\right) \circledast \left((\mathbf{B}^{(n)})^T\mathbf{B}^{(n)}\right)\right]^\dagger \\
&\mathbf{A}^{(n)} \leftarrow \psi\left(\mathbf{A}^{(n)} + \alpha \cdot \mathbf{B}^{(n)}\text{diag}(\boldsymbol{\pi})\mathbf{T}^T\text{diag}(\mathbf{T}\boldsymbol{\pi})^{-1}\right) \\
&\mathbf{B}^{(n)} \leftarrow \mathbf{P}_{(2)}(\mathbf{C}^{(n)} \odot \mathbf{A}^{(n)}) \left[\left((\mathbf{C}^{(n)})^T\mathbf{C}^{(n)}\right) \circledast \left((\mathbf{A}^{(n)})^T\mathbf{A}^{(n)}\right)\right]^\dagger \\
&\mathbf{B}^{(n)} \leftarrow \psi\left(\mathbf{B}^{(n)}\right) \\
&\mathbf{C}^{(n)} \leftarrow \mathbf{P}_{(3)}(\mathbf{B}^{(n)} \odot \mathbf{A}^{(n)}) \left[\left((\mathbf{B}^{(n)})^T\mathbf{B}^{(n)}\right) \circledast \left((\mathbf{A}^{(n)})^T\mathbf{A}^{(n)}\right)\right]^\dagger \\
&\mathbf{C}^{(n)} \leftarrow \psi\left(\mathbf{C}^{(n)} + \alpha \cdot \mathbf{B}^{(n)}\mathbf{T}\right)
\end{aligned}
\tag{3-7}
$$

where $\alpha > 0$ is a scaling factor that decides how much the update equations rely on the ALS update rules or the HMM constraints. Also, note that the factor matrices $\mathbf{B}^{(n)}$ do not depend on $\mathbf{T}$ or $\boldsymbol{\pi}$, causing them not to have a constraint part in the constraint part of the update rules. Just like with the uncoupled CPD update rules, after each calculation, these $\mathbf{A}, \mathbf{B}, \mathbf{C}$ matrices are made sure to be stochastic using $\psi(\cdot)$ from Equation (3-14). These coupled update equations then becomes:

$$
\underset{\{\mathbf{A}^{(n)},\mathbf{B}^{(n)},\mathbf{C}^{(n)}\}_{n=1}^N}{\arg\min} \quad \sum_{n=1}^{N} ||\underline{\mathbf{P}}^{(n)} - [[\mathbf{A}^{(n)}, \mathbf{B}^{(n)}, \mathbf{C}^{(n)}]]||_F^2 \tag{3-8}
$$

s.t.

$$
\begin{aligned}
&\mathbf{A}^{(n)}, \mathbf{B}^{(n)}, \mathbf{C}^{(n)} > 0 \quad \text{(elementwise)} \\
&\sum_{k=1}^{K} \mathbf{A}^{(n)}(:,k) = 1, \quad \sum_{k=1}^{K} \mathbf{B}^{(n)}(:,k) = 1, \quad \sum_{k=1}^{K} \mathbf{C}^{(n)}(:,k) = 1 \\
&\mathbf{A}^{(n)} = \mathbf{B}^{(n)}\text{diag}(\boldsymbol{\pi})\mathbf{T}^T\text{diag}(\mathbf{T}\boldsymbol{\pi})^{-1} \\
&\mathbf{C}^{(n)} = \mathbf{B}^{(n)}\mathbf{T} \\
&\mathbf{T} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{B}^{(n)})^\dagger\mathbf{C}^{(n)} \\
&\boldsymbol{\pi}^T\mathbf{T} = \boldsymbol{\pi}^T
\end{aligned}
$$

For the full derivation, of the minimization problem, as well as a show that the second update steps enfore the HMM constraints for $\mathbf{A}$ and $\mathbf{C}$, please refer to Appendix B-2. $\mathbf{T}$ is found by alternating Equation (3-7) and an averaging of all transition matrices $\mathbf{T}^{(n)}$ which are a solution to $\mathbf{T}^{(n)} = (\mathbf{B}^{(n)})^\dagger\mathbf{C}^{(n)}, \ \forall n \in [1, \ldots, N]$. These transition matrices also need to be permutation matched (Section 3-3-4) before averaging:

$$
\hat{\mathbf{T}} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{B}^{(n)})^\dagger\mathbf{C}^{(n)}. \tag{3-9}
$$

By alternating between solving the extended ALS update rules and finding the common transition matrix $\hat{\mathbf{T}}$, the algorithm finds the estimated HMM matrices along a stable path.

3-2-1 Algorithm

To get a more practical description of Coupled CPD, Algorithm 1 presents the Coupled CPD algorithm in pseudo-code. It intakes N joint probability tensors (JPTs), and the variables K, D, N. It returns the transition matrix $\mathbf{T}$ of the underlying HMM as well as its initial distribution π and all emission matrices $\mathbf{O}^{(n)}$.

Algorithm 1: Coupled CPD-ALS for Multivariate HMM learning

Input : $\underline{\mathbf{P}}^{(n)} \in \mathbb{R}_+^{D\times D\times D} \quad \forall n \in [1, ..., N]$, $K \in \mathbb{Z}$, $D \in \mathbb{Z}$, $N \in \mathbb{Z}$
Output : $\mathbf{T} \in \mathbb{R}_+^{K\times K}$, $\mathbf{O}^{(n)} = \mathbf{B}^{(n)} \in \mathbb{R}_+^{D\times K}$, $\pi \in \mathbb{R}_+^{K}$
Options: $\alpha > 0$, $\rho \in [0, 1]$, maxIter, tolerance, ϵ

Initialization
  Init $\mathbf{A}^{(n)}, \mathbf{B}^{(n)}, \mathbf{C}^{(n)} \quad \forall n = [1, ..., N]$ as $\psi(\mathbf{1}_{D\times K} + \epsilon \cdot \mathcal{U}_{[0,1]})$;
  Init $\mathbf{T}$ as $\psi(\mathbf{I}_K + \epsilon \cdot \mathcal{U}_{[0,1]})$, compute π using Equation (2-1);
  Compute matricizations $\mathbf{P}_{(1)}^{(n)}$, $\mathbf{P}_{(2)}^{(n)}$, $\mathbf{P}_{(3)}^{(n)}$ for each $\underline{\mathbf{P}}^{(n)}$ along modes 1, 2, 3;
Main Loop
  for *iter* $= 1$ *to maxIter* **do**
    $\mathbf{T}_{\text{prev}} \leftarrow \mathbf{T}$;
    for $n = 1$ *to* N **do**
      $\mathbf{A}^{(n)}, \mathbf{B}^{(n)}, \mathbf{C}^{(n)} \leftarrow$ Equation (3-7);
    $\mathbf{T}^{(1)} \leftarrow (\mathbf{B}^{(1)})^{\dagger}\mathbf{C}^{(1)}$;
    for $n = 2$ *to* N **do**
      $\mathbf{T}^{(n)} \leftarrow (\mathbf{B}^{(n)})^{\dagger}\mathbf{C}^{(n)}$;
      $\Pi_{\text{best}} \leftarrow \text{findPermutation}(\mathbf{T}^{(1)}, \mathbf{T}^{(n)})$;
      $\mathbf{T}^{(n)} \leftarrow \Pi_{\text{best}}\mathbf{T}^{(n)}\Pi_{\text{best}}^{T}$;
      $\mathbf{A}^{(n)} \leftarrow \mathbf{A}^{(n)}\Pi_{\text{best}}^{T}$, $\mathbf{B}^{(n)} \leftarrow \mathbf{B}^{(n)}\Pi_{\text{best}}^{T}$, $\mathbf{C}^{(n)} \leftarrow \mathbf{C}^{(n)}\Pi_{\text{best}}^{T}$;
    $\mathbf{T} \leftarrow \rho \cdot \mathbf{T}_{\text{prev}} + \frac{1-\rho}{N}\sum_{n=1}^{N}\mathbf{T}^{(n)}$;
    Compute π using Equation (2-1);
    if $||\mathbf{T} - \mathbf{T}_{prev}||_F < tolerance$ **then**
      Break loop;

3-3 Considerations, relevant measures and functions

3-3-1 Influence of sampling rate

When fitting a Hidden Markov Model (HMM) to real data, the transition matrix $\mathbf{T}$ depends on the sampling rate f_s. As the sampling rate decreases, the transition probability of staying in the same state becomes smaller, and the probability of transitioning to a different state becomes higher. This means the transition matrix becomes more uniformly distributed. As the sampling rate increases, the probability of staying in the same state over time becomes larger, and the probability of transitioning to a different state becomes smaller. This relates

to the transition matrix approaching an identity matrix:

$$\begin{aligned} \lim_{f_s \to 0} \mathbf{T}(f_s) &= \frac{1}{K} \mathbf{1}_K \mathbf{1}_K^T, \\ \lim_{f_s \to \infty} \mathbf{T}(f_s) &= \mathbf{I}_K, \end{aligned} \tag{3-10}$$

where $\mathbf{1}_K$ is a column vector of length K filled with ones, and $\mathbf{I}_K$ is an identity matrix of size $K \times K$. It could be argued that if the sampling rate is too low, the HMM does not capture the probabilistic behavior. As can be seen in Figure 3-4, even though State 3 was active roughly halfway through the state sequence, the low sampling rate caused this behavior to be missed.

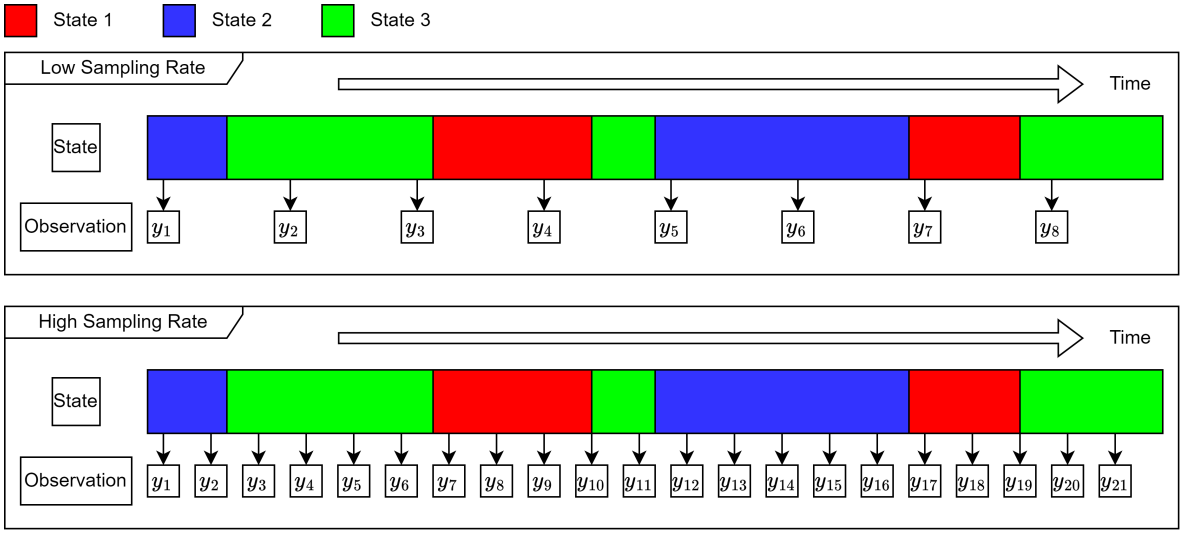


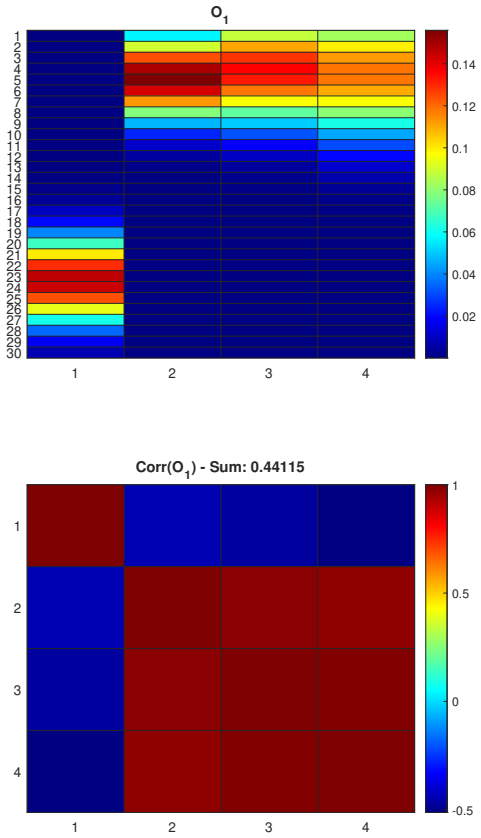
Figure 3-4: A diagram showing the influence of the sampling rate on the probability of transitioning to the same state.

3-3-2 Emission correlation measure

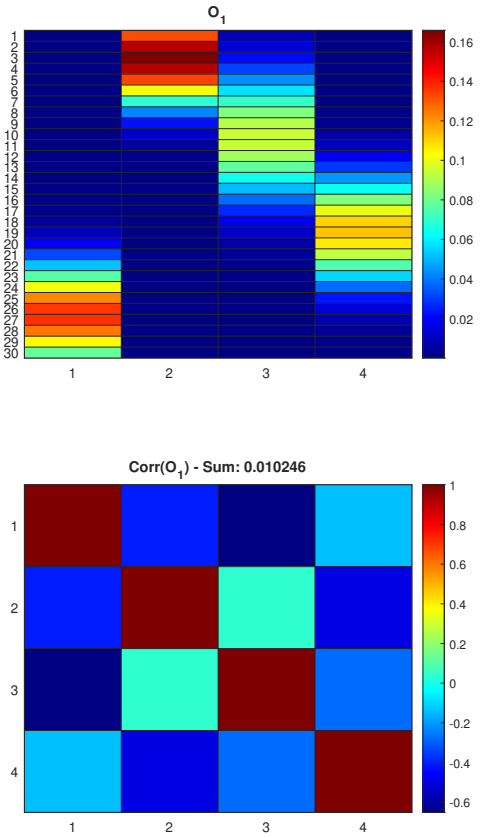
Canonical Polyadic Decomposition (CPD) needs the columns of the emission matrix to differ from each other to obtain good results. Of course, this includes that the matrices need to have a Krushkal rank of at least K, but there is also a need for distinct features in each of the columns, making it a more general requirement. This is also true for an HMM; if two states have the same emission columns and thus distributions, they might as well be the same state, as the difference in distribution is the only distinctive feature that makes states differ from each other. The question then becomes how similar the columns can be for the CPD to still perform properly. To check how similar two column vectors are, a correlation measure can be used:

$$\text{corr}(X, Y) = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y} = \frac{\mathbb{E}[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}. \tag{3-11}$$

For the emission matrix $\mathbf{O} \in \mathbb{R}^{D \times K}$ the correlation between the columns or distributions can be compactly expressed as $\text{corr}(\mathbf{O})_{ij} = \text{corr}(\mathbf{O}_{:,i}, \mathbf{O}_{:,j})$, which results in a $K \times K$ correlation matrix $\text{corr}(\mathbf{O})$. Two examples of emission matrices and their correlation matrix can be found in Figure 3-5. Then, using this matrix, a single value can be extracted to show how



(a) Emission matrix initialized without mean offset and its correlation matrix.



(b) Emission matrix initialized with mean offset and its correlation matrix.

Figure 3-5: An example of a difference between induced offset in the emission distribution means (in Section 4-1-1 named offset) and no such offset (in Section 4-1-1 named random), with an effect on the respective correlation matrices and correlation numbers.

uncorrelated or independent the columns are from each other by summing up all elements of the correlation matrix. To ensure independence of this number from the number of states, it can be divided by K^2:

$$\phi(\mathbf{O}) = \frac{\sum_{i,j}^{K} \text{corr}(\mathbf{O})_{ij}}{K^2}. \tag{3-12}$$

Here $\text{corr}(\mathbf{O})$ has a theoretical maximum of K^2 if all columns of $\mathbf{O}$ are the same, giving $\phi(\mathbf{O})$ a theoretical maximum of 1. Since all entries of $\mathbf{O}$ will be positive, no anticorrelation can take place, making the theoretical minimum of both $\text{corr}(\mathbf{O})$ and $\phi(\mathbf{O})$ equal to 0.

3-3-3 Error metric for HMMs

To measure how good an estimation the found HMM matrices are to the original/ground truth, an error metric needs to be introduced. The error is measured between the transition matrices and the emission matrix. For the multivariate case, it is a measure between all emission matrices; however, as the score should not become larger for the number of time series or emission matrices, which will make error comparisons between different N's impossible,

the cost for the emission matrices is averaged over N:

$$C(\mathbf{T}, \mathbf{O}, \hat{\mathbf{T}}, \hat{\mathbf{O}}) = \frac{||\mathbf{T} - \hat{\mathbf{T}}||_F}{||\mathbf{T}||_F} + \frac{1}{N} \sum_{n=1}^{N} \frac{||\mathbf{O}^{(n)} - \hat{\mathbf{O}}^{(n)}||_F}{||\mathbf{O}^{(n)}||_F}. \tag{3-13}$$

As the initial distribution $\boldsymbol{\pi}$ can be calculated through the transition matrix $\mathbf{T}$, given that the HMM is ergodic, it is not included in the error metric.

3-3-4 Scaling and Permutation Ambiguity

Scaling Ambiguity

CPD and other decomposition methods generally suffer from both scaling and permutation ambiguity. Scaling ambiguity is easily solved, as all columns of the HMM matrices are known to be stochastic, meaning all entries are positive and the elements of the columns sum up to one, which can be enforced on the processed factor matrices after the fact. The column stochasticity should hold for the CPD factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$, the transition matrix $\mathbf{T}$, the emission matrices $\mathbf{O}^{(n)}$, and the initial distribution $\boldsymbol{\pi}$. To enforce column stochasticity, the following can be used:

$$\psi(\mathbf{M}_{:,k}) = \frac{|\mathbf{M}_{:,k}|}{\sum |\mathbf{M}_{:,k}|}, \ \ \forall\, k \in \{1, \ldots, K\}. \tag{3-14}$$

Permutation Ambiguity

Permutation ambiguity, however, can only be solved concerning a ground truth or some arbitrary permutation, and even then, it is a difficult problem. When working with real data, the order of the states and thus permutation ambiguity is not important concerning a ground truth, as there is none. It is important that the permutation of the transition matrix and emission matrix match, as well as the permutations between emission matrices in the multivariate case. When working with synthetic data, a permutation comparison can be made between the ground-truth HMM matrices $\mathbf{T}, \mathbf{O}$ and the estimated matrices $\hat{\mathbf{T}}, \hat{\mathbf{O}}$. If the estimated matrices are a close match (still having permutation ambiguity) to the ground truth, the permutation can be removed by a postprocessing algorithm. When the error between the estimates and the ground truth matrices is large, it is hard to find a matching permutation for the estimates. A flowchart of the permutation mismatch can be observed in Figure 3-6. For multivariate HMMs with N time sequences, there will be N Joint Probability Tensors (JPTs) and, thus, N emission matrices. When applying CPD to obtain the factor matrices for each JPT, there is no guarantee that these factor matrices have the same permutation. As each emission matrix for all N time sequences will differ, so will all three of the associated factor matrices; they should, however, have a common underlying transition matrix $\mathbf{T}$. When all factor matrices suffer from permutation ambiguity between different N JPTs, the different found $\mathbf{T}$ matrices also suffer from permutation ambiguity concerning each other. As all transition matrices are supposed to be the same, however, the permutation ambiguity can be removed by comparing these. This does mean that the transition matrices $\mathbf{T}^{(n)}$ should be similar enough (still suffering from permutation ambiguity).

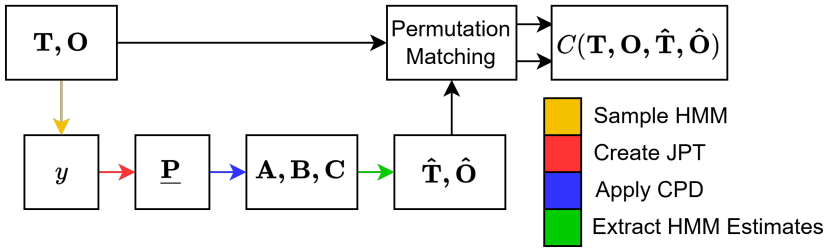


Figure 3-6: A diagram that shows using known HMM matrices $(\mathbf{T}, \mathbf{O})$ to create synthetic data y. This synthetic data is then used to create a JPT, which is then used to estimate the HMM matrices. The permutation ambiguity in these estimates needs to be removed to relate the estimate to known matrices through an error metric.

To remove the permutation ambiguity between the transition matrices $\mathbf{T}^{(n)}$, first, a reference is needed. With synthetic data, this can be the ground truth, and with real data, this can be the transition matrix of the first data sequence $\mathbf{T}^{(1)}$. Then the most obvious algorithm is to check all possible permutations of matrix $\mathbf{T}_B$ through permutation matrix $\boldsymbol{\pi}$ and find the best fit to matrix $\mathbf{T}_A$ by checking the difference $D = ||\mathbf{T}_A - \boldsymbol{\pi}\mathbf{T}_B\boldsymbol{\pi}^T||_F$. Then the permutation that results in the smallest D is the best. This scheme requires lots of checks as the number of possible permutations is $K!$, and thus is very expensive for large K. To combat this, one could implement a heuristic algorithm. In this report, another approach is applied first.

This initial approach consists of comparing the diagonal entries of the two transition matrices $\mathbf{T}_A$ and $\mathbf{T}_B$. This is done by looping over all K diagonal entries of $\mathbf{T}_B$ and finding the best match in diagonal entries of $\mathbf{T}_A$. If no two best matches are the same, a unique permutation is found, which can then be said to be the best match. This is possible because the transition matrices are known to be close to identity, making the diagonal entries the most influential. This initial approach is not guaranteed to find a solution, and if it fails, a brute force method is still applied. This combination results in Algorithm 2.

3-3-5 Signal-to-Noise Ratio (SNR)

The Signal-to-Noise Ratio (SNR) is a commonly used descriptor for how noisy data is, and is given by:

$$\text{SNR} = \frac{\mathbb{E}(S^2)}{\mathbb{E}(N^2)},$$

where S is the signal and N is the noise. Through the formula for variance, the second uncentered moments above can be found through the formula for the second centered moment, the variance:

$$Var(X) = \mathbb{E}(X^2) - (\mathbb{E}(X))^2 \quad \rightarrow \quad \mathbb{E}(X^2) = Var(X) + (\mathbb{E}(X))^2.$$

As the noise is zero-mean white noise, this results in the following SNR:

$$SNR = \frac{\sigma_S^2 + \mu_S^2}{\sigma_N^2}.$$

Note that for the creation of synthetic data, the SNR can be controlled by controlling the variance of the noise σ_N^2. Note also that the mean of the signal has an effect on the SNR,

but the values associated with the bins of the emission matrix are arbitrary. Therefore, when working with continuous data, which is binned for discrete HMMs (dHMMS), the data needs to be normalized first between -1 and 1, keeping the signal mean at 0. This normalization also needs to be taken into account when calculating the SNR. As the mean of the signal becomes zero, the SNR becomes purely a measure between the variance of the signal to the variance of the noise.

3-3-6 Initialization of $\mathbf{T}, \mathbf{A}^{(n)}, \mathbf{B}^{(n)}, \mathbf{C}^{(n)}$

For the coupled CPD for multivariate HMM learning, the following matrices have to be initialized: $\mathbf{T}, \mathbf{A}^{(n)}, \mathbf{B}^{(n)}, \mathbf{C}^{(n)}, \ \forall n \in [1, \ldots, N]$. It is known that the transition matrix usually shows a similar structure to an identity matrix with some noise added. This means the transition matrix can be initialized as such at the start of the algorithm. The factor matrices, however, have a structure that depends on the distributions of the emissions, which are not known and therefore cannot be given prior information. An assumption could be made about the distributions being Gaussian, but even then, it is not known what the mean and standard deviation are associated with what state. Therefore, the factor matrices are initialized uniformly, with noise added. The noise added to both the transition matrix and factor matrices is important, as fully uniform and identity matrices will result in no convergence at all:

$$\mathbf{T}_{init} = \psi(\mathbf{I}_K + \epsilon \cdot \mathcal{U}_{[0,1]}), \qquad \{\mathbf{A}^{(n)}, \mathbf{B}^{(n)}, \mathbf{C}^{(n)}\}_{n=1}^{N} = \psi(\mathbf{1}_{D\times K} + \epsilon \cdot \mathcal{U}_{[0,1]}).$$

3-3-7 Computational complexity

In this section, the largest consumers of computational cost with regard to this method are discussed. Some other parts of the process are ignored as their computational cost is negligible in comparison to the rest of the process, such as:

- The extended part of the Alternating Least Squares (ALS) update equations;
- Making the factor matrices stochastic every iteration;
- Calculating the HMM matrices $\mathbf{T}^{(n)}, \mathbf{O}^{(n)}, \boldsymbol{\pi}^{(n)}$ given the factor matrices $\mathbf{A}^{(n)}, \mathbf{B}^{(n)}, \mathbf{C}^{(n)}$;
- Permuting the factor matrices and transition matrix, given that the optimal permutation has already been found;
- Averaging the transition matrices $\hat{\mathbf{T}}^{(n)}$ to obtain $\hat{\mathbf{T}}$;
- Updating the transition matrix each iteration.

JPT creation

The creation of JPTs depends on the number of samples M and the number of data sequences N. Both of which have a linear effect on the computational cost. Meaning that JPT creation has a computational complexity of $\mathcal{O}(NM)$.

Removal of Permutation Ambiguity

Permutation ambiguity occurs in the columns of the factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ (so also in the emission matrix), and in the rows and columns of the transition matrix $\mathbf{T}$. As the initial distribution is calculated from the transition matrix $\mathbf{T}$, by assuming ergodicity, its permutation ambiguity can be ignored. All matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\mathbf{T}$ suffer from the same permutation ambiguity, so a single permutation needs to be found. As the factor matrices of different data sequences/JPTs cannot be compared with each other, as they can be completely different, the permutation removal needs to be applied to the transition matrices. This means that they need to be sufficiently similar.

The permutation cost can be checked by checking the Frobenius norm of the difference between the transition matrix $\mathbf{A}$ and the permuted transition matrix $\mathbf{B}$. If the solution is brute forced, the computational complexity is $\mathcal{O}(K!)$. One can apply a trick where the diagonal entries (dominant entries in a transition matrix) of one transition matrix A, find a best match in the diagonal entries of transition matrix B. This has a computational complexity of $\mathcal{O}(K)$, but has a way lower computational cost per entry concerning brute forcing, as that requires expensive matrix calculations. The extra time cost of brute forcing is described here with $c \gg 1$. This trick is not always guaranteed to find a solution; however, in which case, the calculation is still brute forced. This results in a minimal computational complexity of $\mathcal{O}(K)$ and a maximum of $\mathcal{O}(K + (c \cdot K)!)$. The full algorithm of removing permutation ambiguity is found in Appendix A.

Coupled CPD

When decomposing a JPT, which is $D \times D \times D$ with rank K, it has a computational complexity of $\mathcal{O}(D^3K)$. When running this for all sequences, the complexity increases linearly with N, resulting in $\mathcal{O}(ND^3K)$. The question then is, does the coupling induce extra computational cost? For the extended ALS update rules, the HMM soft constraint part introduces negligible extra computation time. However, before calculating the average of the transition matrices, the permutation needs to be removed for $N-1$ transition matrices, and as discussed before, this is quite expensive, also given that this is done each iteration. This finally results in $\mathcal{O}(ND^3K + K(N-1))$ as a minimum and $\mathcal{O}(ND^3K + (K + (c \cdot K)!)(N-1))$ as a maximum.

3-4 Summary

This chapter outlines a method for learning Hidden Markov Models (HMMs) using Canonical Polyadic Decomposition (CPD), focusing on transitioning from univariate to multivariate settings. Initially, univariate HMMs are learned through CPD, and then these are extended to multivariate problems by coupling them with a common transition matrix. The Alternating Least Squares (ALS) method is used for minimization, with enhancements to address permutation and scaling ambiguities. Performance metrics are discussed as well as computational complexities, highlighting permutation ambiguity as a significant computational cost.

Chapter 4 presents the simulation results of the proposed method along with a discussion based on the performance metrics introduced in this chapter. The algorithm is compared to both Baum-Welch and Uncoupled CPD.

Chapter 4

Simulation Results and Discussion

This chapter explores the behavior of the proposed method, multivariate Hidden Markov Model (HMM) learning through Coupled Canonical Polyadic Decomposition (CPD). The method is tested with various performance variables and compared to the current standard, the Baum-Welch algorithm.

Tests on discrete synthetic data can be found in Section 4-1, where data is generated from a discrete HMM (dHMM). Different options for ground truth HMMs are compared based on different distributions of the transition matrix and correlations in the distributions of the mission matrices, and the response of the proposed method over these is analyzed. The performance of the algorithm is also evaluated based on a varying number of states K, emissions D, data sequences N, and samples m. The goal is to determine the optimal operating ranges for the HMM parameters with respect to this algorithm and to identify situations where it might be more advantageous to revert to the Baum-Welch algorithm. Additionally, the algorithm's performance is assessed concerning the parameters α and ρ. This analysis aims to reveal the effects of these variables on linking the separate data sequences through a common transition matrix $\mathbf{T}$. Lastly, all parameters are chosen favorably, and the performance of Baum-Welch, Uncoupled CPD, and Coupled CPD are compared with respect to HMM cost and computation time.

Simulations are also performed in continuous synthetic data generated from a continuous HMM (cHMM), which enables the addition of noise to the generated sequence data. This allows the observation of the effect of noise on the performance of the proposed method, as discussed in Section 4-2.

Finally, the method is applied to real data, namely sleep stage data, which can be found in Section 4-3. Here the goal is to learn HMM matrices from the data and use the Viterbi algorithm to find the most likely path of states that lie underneath the observed data. This path is then compared with the annotations of sleep stages that come with the data and is done by experts to verify the accuracy of the proposed method.

4-1 Discrete synthetic data

Discrete synthetic data refers to data generated by sampling from a discrete Hidden Markov Model (dHMM). Initially, a state is chosen based on the initial distribution $\boldsymbol{\pi}$. For each time step, a new state K is selected using the transition matrix $\mathbf{T}$. An emission $o_t \in [1, \ldots, D]$ is then determined for each data sequence $n \in [1, \ldots, N]$, based on a column from the emission matrix $\mathbf{O}^{(n)}_{:,k}$ that corresponds to the previously mentioned state K and data sequence n. This process results in N strings of pre-binned data, which consist of integers ranging from $[1, \ldots, D]$, as illustrated in Figure 4-1.

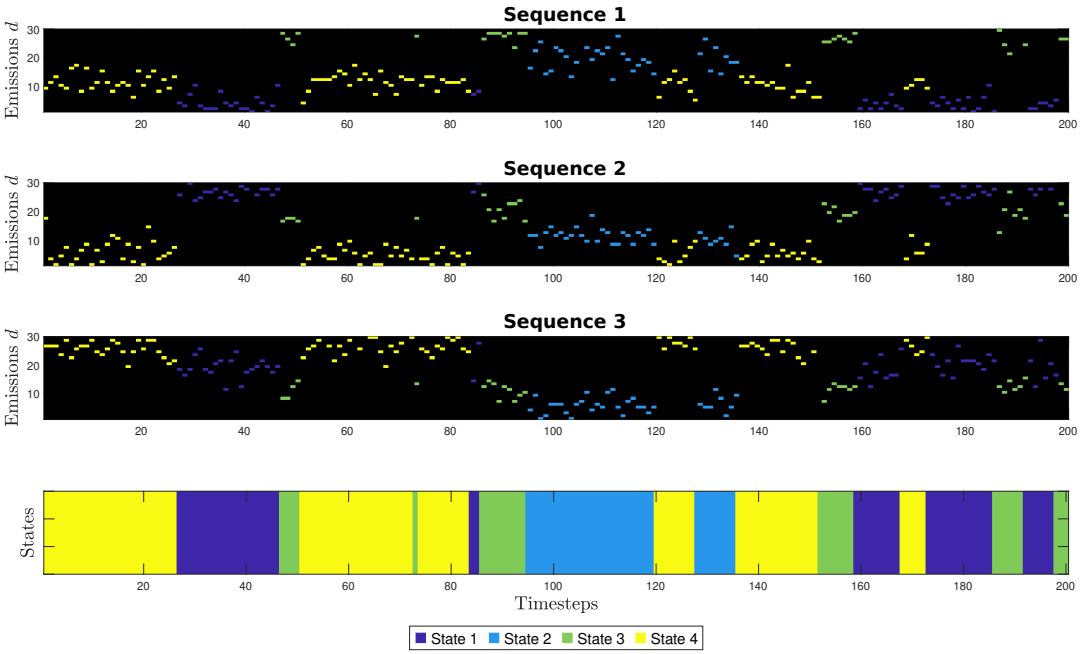


Figure 4-1: An example of discrete data from a dHMM for $K = 4, D = 30, N = 3$, and $m = 200$. The underlying active states and the data sequences that are a result of the emissions of those states can be observed.

This section aims to analyze the behavior of the proposed method in relation to discrete data. To achieve this, the methods are tested with varying numbers of states K, emissions D, and observed sequences N. Additionally, changes in the underlying ground truth HMM matrices $\mathbf{T}$ and $\mathbf{O}^{(n)}$ are examined, as well as the hyperparameters α and ρ. Finally, the impact of similarity or correlation in the distributions of states $\mathbf{O}_{:,k}$ for all $k \in [1, \ldots, K]$ is discussed.

4-1-1 Variation in ground truth HMM matrices

To assess the performance of Coupled Canonical Polyadic Decomposition (CPD), variations in the ground truth matrices are analyzed. This indicates the types of HMMs for which Coupled CPD is effective in uncovering, as well as those for which it is not.

Transition matrix

For simulation purposes and synthetic data, a transition matrix is used that is based on an identity matrix in combination with noise. Hence, $\mathbf{T}$ will be initialized as follows:

$$\mathbf{T} = \psi\left(\mathbf{I}_K + \epsilon\, \mathcal{U}_{[0,1]}\right), \tag{4-1}$$

where $\mathbf{I}_K$ is an identity matrix of size $K \times K$, ϵ is a small number, and $\mathcal{U}_{[0,1]}$ is a uniform distribution with bounds 0 and 1.

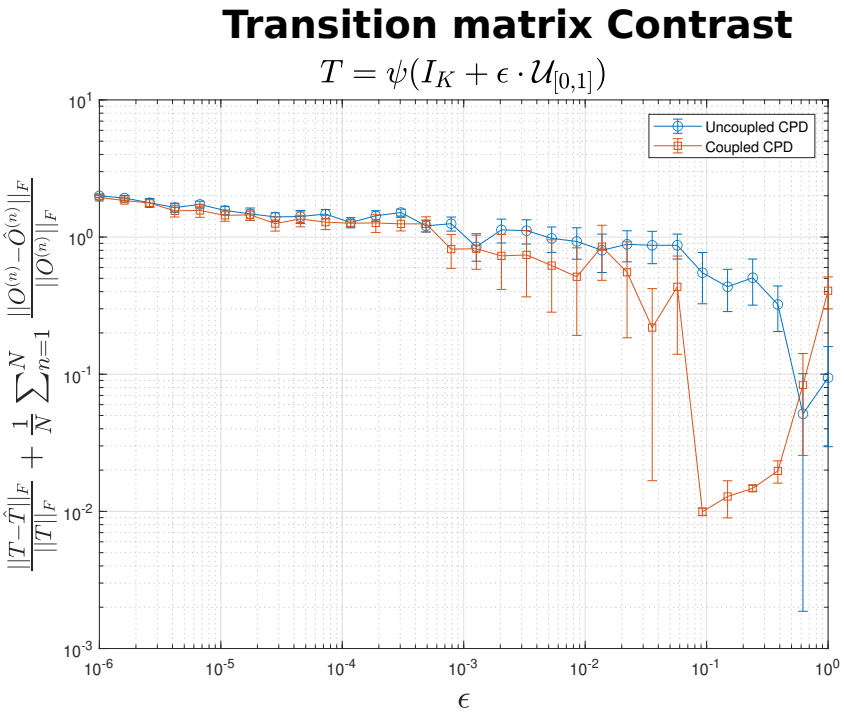


Figure 4-2: A plot showing the performance of HMM estimation over the variable ϵ, which controls the contrast in the ground truth transition matrix. In this plot, the following variables are fixed: $K = 4, D = 30, N = 5, \alpha = 0.1, \rho = 0.85, m = 10^5$. Each data point is the mean of 10 runs, and shows the standard deviation over those runs through the bars on each data point.

In Figure 4-2, the error metric of learning HMMs can be observed over ϵ, a variable that controls the contrast of probabilities in the ground truth transition matrix. Each data point in the plot is an average of 5 different runs, and all runs have different initializations for the ground truth matrices. It can be observed that the HMM learning operates best when $\epsilon \approx [0.1, \ldots, 0.2]$. This can be explained, from a CPD perspective, by the fact that if ϵ is very small, the identity part becomes dominant, leaving little to no difference in the diagonal entries of $\mathbf{T}$. This creates difficulty for the algorithm to remove permutation ambiguity, especially during convergence. From an HMM perspective, if the diagonals of the transition matrix $\mathbf{T}$ become more dominant, there is a smaller chance of transferring to a different state. When $\epsilon < 10^{-5}$ with $m = 10^5$, it might even occur that a particular state is never entered in the whole data sequence, making it hard to learn the transition matrix $\mathbf{T}$.

When ϵ is relatively large, $\epsilon \approx 1$, the transition matrix becomes close to being uniformly distributed, meaning a clear distinct structure disappears. This also results in the $\mathbf{A} \sim P(y_{t-1}|x_t)$ and $\mathbf{C} \sim P(y_{t+1}|x_t)$ factor matrices, which are based on the transition matrix $\mathbf{T}$ and emission matrix $\mathbf{O}$, to have flatter distributions along the columns. This removes distinct features in the distribution of the previous and next observation, given the current state, making it harder for the algorithm to capture a clear relationship with the probability of the current observation given the current state.

Emission matrix - distribution correlation influence

The emission matrices used as ground truth have Gaussian distributions along the columns, as an example. In reality, any Probability Mass Function (PMF) could be used as a distribution, which can be learned as long as its features are distinct enough from PMFs of different states within the same data sequence. To show the performance of Coupled CPD for variations in the emission matrices, in this section, the correlation between these emissions will be taken into account, like in Section 3-3-2.

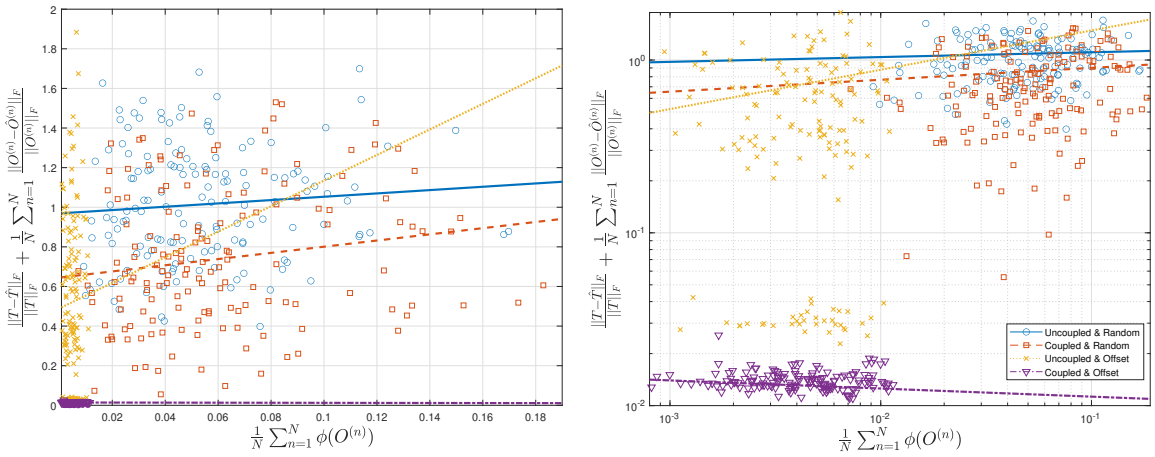
When two emissions from two states in an HMM are identical, there is no way to distinguish those two states, because the emission probability distribution is the only characterizing factor for HMM states. Similarly, CPD has the Krushkal-rank condition for uniqueness, stating that two columns of the factor matrices cannot be the same; they need to be linearly independent to find a unique solution. Linear independence is a strict constraint, as any small perturbation of one of the two similar vectors can cause them to be independent of each other. In CPD HMM learning, however, it becomes apparent that small differences between columns can still return bad results/runs. This means there is a need for a more general or loose constraint. Because of this, this section looks at the correlation between the different columns in emission matrices, and thus at similarities between distributions of different states. The main goal is to observe how this correlation in the emission distribution of ground truth HMM matrices affects the algorithm's ability to accurately estimate these matrices.

In Figure 4-3, the HMM cost defined in Equation (3-13), over the average correlation between columns of emission matrices of all N sequences, or Joint Probability Tensors (JPTs), can be observed. There are four data clouds plotted. Coupled CPD and Uncoupled CPD, and ground truth emission matrices initialized with random means for the normal distributions (random), and initialized with an offset (offset) for the mean of each distribution.

Note that both the Uncoupled method and the Coupled method respond better to an offset in the mean of the distributions. This is because it makes the distributions of each state more easily distinguishable. The Coupled method, however, outperforms the Uncoupled method. The error of Coupled CPD is generally lower for the same average correlation between distributions.

4-1-2 Performance over K, D, N and m

This section tests the performance of a multivariate Baum-Welch algorithm [24, 25, 26] as a reference and literature standard, of an Uncoupled CPD method as described in Section 3-1, and the Coupled CPD method, which is the novel approach of this report. The performance



(a) Linear scale **(b)** Log scale

Figure 4-3: A scatterplot of the HMM cost plotted over the correlation between columns of $\mathbf{O}$ averaged over all $N = 5$ sequences. Each scatter point is a separate run of which there are 150 per cluster. Four clusters can be observed for Uncoupled and Coupled CPD learning, with induced offsets in the mean of the distributions (offset) and no such induced offset (random). Other parameters for this plot include: $K = 4, D = 30, m = 10^5, \alpha = 0.1, \rho = 0.85$.

of these algorithms is tested along four important variables for HMMs: the number of states of the ground truth K; the number of emissions D used by Uncoupled CPD and Coupled CPD; the number of sequences N used in the multivariate dataset; and the number of samples m in each data sequence.

Number of states K

The number of states in HMMs affects not only the computation time required for learning algorithms but also the accuracy of the model in relation to the ground truth. From the perspective of Coupled CPD, an increase in the number of states leads to a rise in the number of columns in the factor matrices as well as the rank of the decomposed tensor. A higher rank in the tensor increases the likelihood of overfitting to noise and can result in computational instability, as noted in [27].

When the number of emissions remains constant while the number of states increases, from an HMM perspective, it becomes more challenging to differentiate between the emissions associated with those states. Although the emission matrix gains additional columns, the number of rows remains unchanged. Since all values in the matrix are positive, this condition heightens the correlation between the columns, as they are more likely to overlap. Consequently, this leads to a higher reconstruction error due to the lack of distinctiveness.

The impact of this issue is reduced if the emissions for different states are sufficiently distinguishable. For Gaussian distributions, this implies having a low variance (σ^2), indicating distinct peaks, and sufficiently different means (μ) for the emissions of different states. Additionally, increasing the number of emissions can help accommodate a higher number of states, thereby enhancing the discretization resolution of the data sequences.

In summary, as the number of states increases, the error associated with HMMs is expected to rise as well. Furthermore, the computation time is anticipated to increase significantly for Coupled CPD as the number of states grows, primarily due to the complexity of the permutation ambiguity removal algorithm, which scales with $K!$.

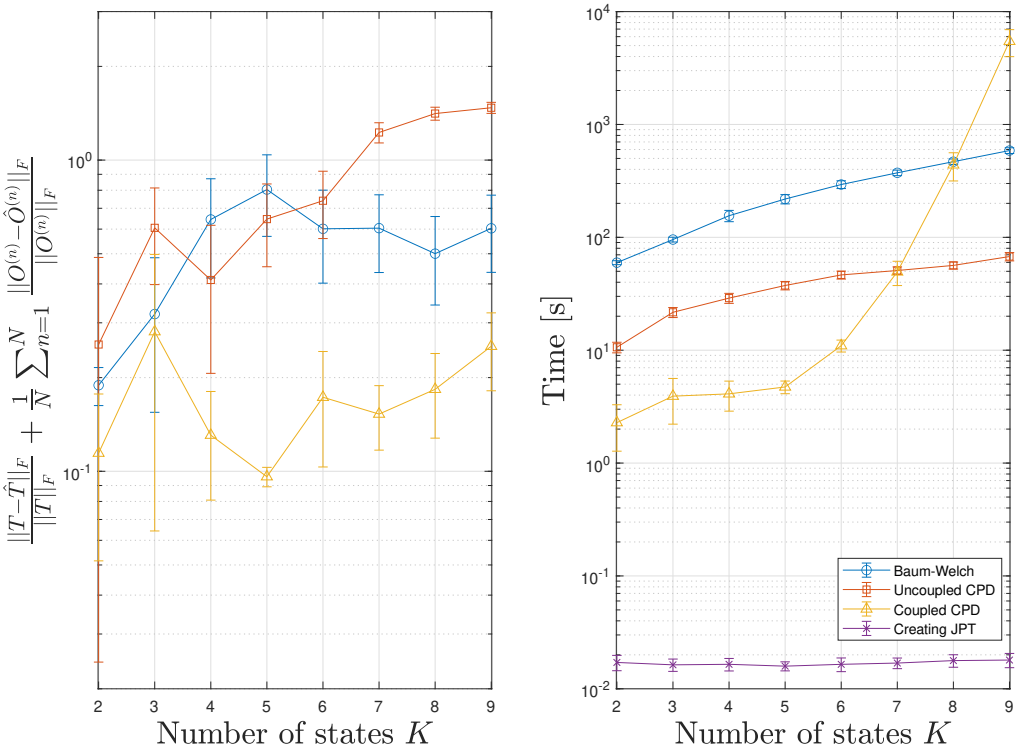


Figure 4-4: A plot that shows the HMM error (left) and time cost (right) when the numbers of states increase. Note that each point on this plot uses a different HMM, and the runs are averaged over 20 runs. Also, the standard deviation can be observed through the bars at each point, such that 66% of occurrences fall within the bars, without regard for skewness (see Figure C-2 for a boxplot of the same data for more exact statistical details). The following parameters are used: $D = 30, N = 5, m = 10^5, \alpha = 0.05, \rho = 0.85$.

Cost plot - Figure 4-4 - The HMM error can be observed for the Baum-Welch algorithm, Uncoupled CPD, and Coupled CPD. It can be seen that the Baum-Welch increases in error from $K = 2$ to $K = 5$ after which it decreases. From the box plot, Figure C-2, the variance can be observed to take a jump at 4 states, after which it slowly decreases.

Uncoupled CPD can be seen to increase in HMM error as the number of states grows, which is due to an increase in distributions without increasing the discretization resolution (number of emissions D). When looking at the box plot, Figure C-2, One can observe something common, Uncoupled CPD can occasionally outperform the other methods, but is not stable enough to do so consistently.

Coupled CPD seems to vary significantly in standard deviation and average cost; however, due to the plot being log scale, it is on par with variations in the other two methods. The

error can, however, be observed to be smaller than for the other two methods, which is due to a coupling of all data sequences through a common transition matrix. This constraint keeps the relationship between the factor matrices $\mathbf{A} = P(y_{t-1}|x_t), \mathbf{B} = P(y_t|x_t), \mathbf{C} = P(y_{t+1}|x_t)$ as they should be, whilst, in Baum-Welch and Uncoupled CPD, these are allowed to vary or be influenced by the underlying cost function manifold. From the box plot, also quite a few outliers can be noted, which (when compared to the other box plots) is more than usual, and can be attributed to only having 20 runs to average on.

Time plot - Figure 4-4 - With the number of emissions D and the sequence length m staying constant, so can the calculation time of the JPTs be observed to stay constant. For the calculation of a JPT, the number of states in the underlying distribution has no influence.

For Baum-Welch, on the other hand, the calculation time can be observed to increase as the number of states does; this increase can be seen to be very predictable, as the standard deviation of this time cost is very small.

For Uncoupled CPD, the same can be observed as for Baum-Welch, with the exception that the method is nearly an order of magnitude faster.

The Coupled CPD method, however, increases enormously in calculation time. This was already mentioned in Section 3-3-7 and is due to the permutation matching algorithm, which has a $\mathcal{O}(K!)$ influence on the calculation time. It will be advised to use a different method to solve the permutation ambiguity problem, as by all other measures, Coupled CPD generally outperforms the other two methods in calculation time due to its fast convergence.

Due to low standard deviations for all methods regarding calculation time, the box plot in Figure C-2 tells the same story.

Summary - Coupled CPD can be observed to generally outperform both other methods concerning the HMM cost. For the time cost, Coupled CPD is costly for a large number of states due to the permutation ambiguity removal algorithm, and it is advised to use another algorithm (heuristic) instead.

Number of possible emissions D

An increase in the number of possible emissions, and thus the (value) discretization resolution of the data, can be associated with more accurately describing the data. One would expect the reconstruction cost to go down, and of course calculation time to go up. However, when considering a JPT, the number of samples m must also increase as the number of possible emissions D does. The number of samples has an important influence on how accurately the data can describe the true underlying distribution. Therefore, just increasing D is meaningless. In this subsection, $m = 20 \cdot D^3$, such that it scales with the dimensions of the JPTs. The calculation time is expected to increase with the number of emissions as it has a $\mathcal{O}(D^3)$ influence on computational complexity. Note that Baum-Welch is not included as it is a multivariate continuous method, and will therefore not be influenced by D.

Cost plot - Figure 4-5 - The cost of Uncoupled CPD can be observed to remain relatively constant in mean with a large standard deviation, suggesting large variations in the cost. This is also confirmed by the box plot Figure C-3, where the median can be observed to be rather stationary, but the variations are significant.

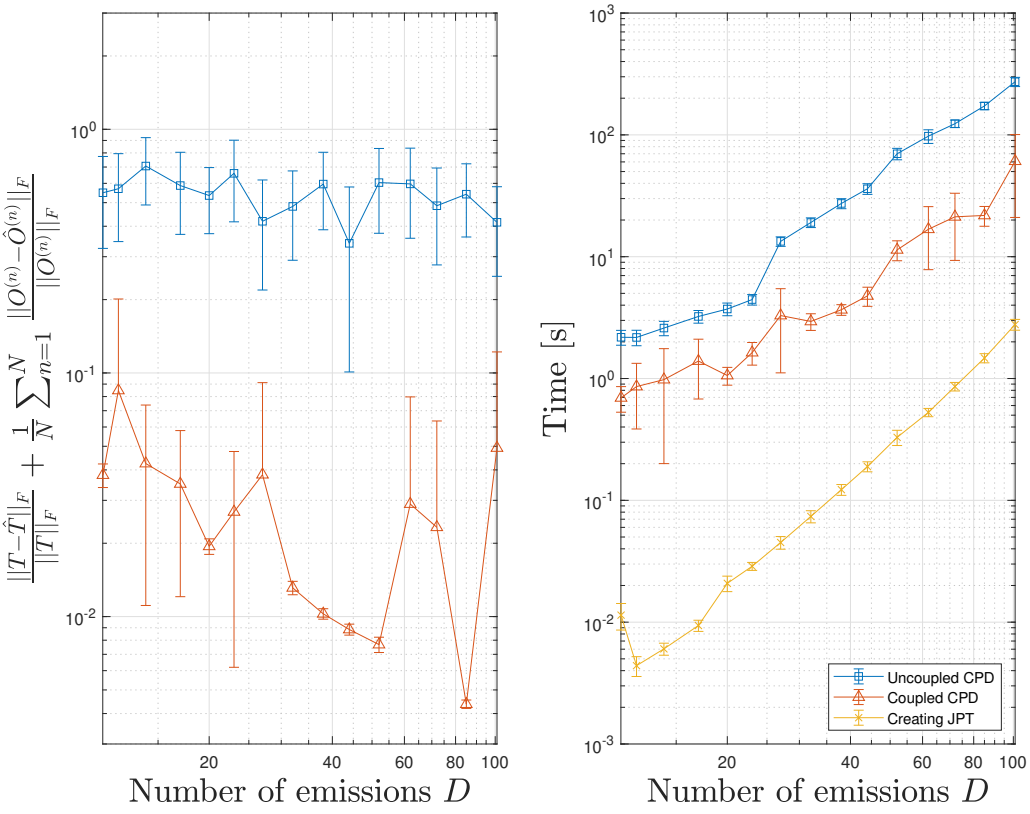


Figure 4-5: A plot that shows the HMM error (circles) and time cost (triangles) when the number of possible emissions increases. Each data point is the mean of 20 runs and shows the standard deviation of this average through the bars attached to each point. The following parameters $K = 4, N = 5, \alpha = 0.05, \rho = 0.85$ are used in this plot. As the number of emissions D increases, so does m need to increase to avoid numerical instability, as such $m = 20 \cdot D^3$.

Coupled CPD also seems to vary a lot in both mean and standard deviation. However, when looking at the box plot, Figure C-3, it can be observed that this behavior is due to a few large outliers. The non-outliers show a convergence of the HMM error as D increases. Also, note again that, as this plot is on a log scale, the mean and standard deviation for Coupled CPD intuitively appear more extreme than those of Uncoupled CPD, which is not the case.

Time plot - Figure 4-5 - The time duration for both Uncoupled and Coupled CPD increases as D increases, as expected. The fact that Coupled CPD has a smaller calculation time than Uncoupled CPD can be attributed to fewer iterations until convergence, not to less time per iteration.

Summary - Coupled CPD can be observed to respond better to an increase in D than Uncoupled CPD when it comes to HMM cost (except for a few outliers) and calculation time.

Number of data sequences N

An increase in the number of data sequences, and thus the number of JPTs, has a linear effect on both the time it takes to create the JPT and the time it takes to run the Coupled CPD scheme. For the creation of the JPT, the computational complexity was expected to be $\mathcal{O}(Nm)$, which showcases the linear dependence. For both the update rules of Uncoupled CPD and Coupled CPD, N is expected to have a linearly increasing effect. However, on Coupled CPD, N also affects the permutation matching algorithm, which is computationally expensive, because of this, it is expected to scale worse on calculation time.

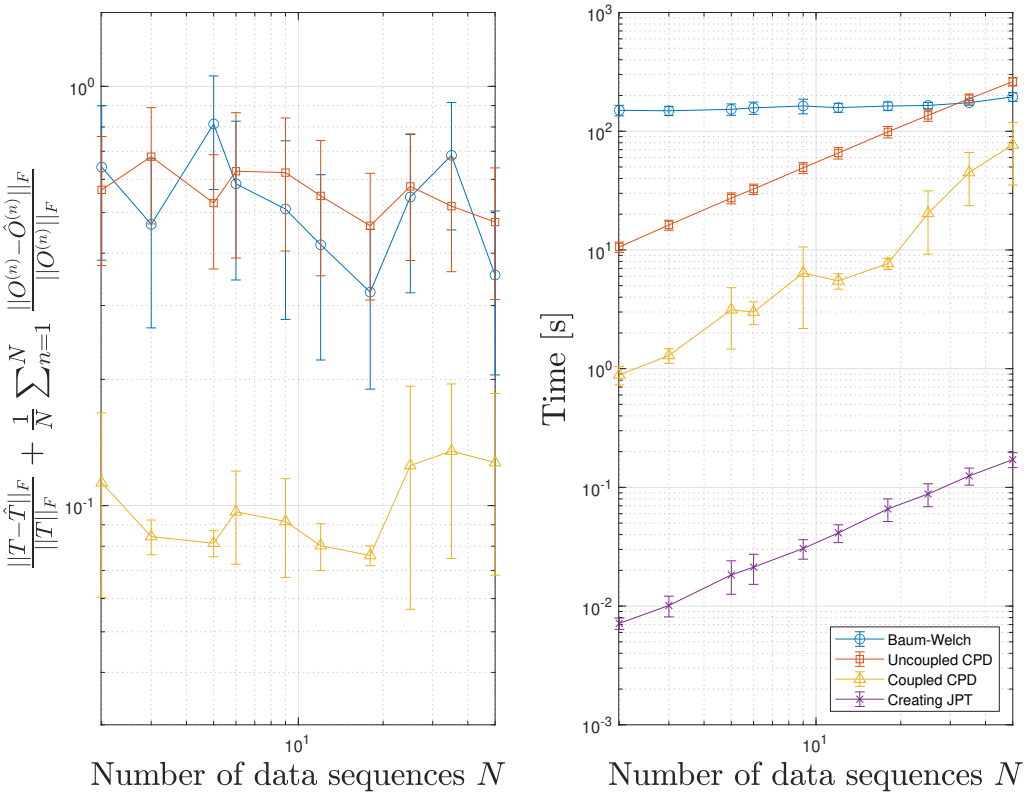


Figure 4-6: A plot that shows the HMM error (circles) and time cost (triangles) when the number of data sequences, and thus the number of JPTs, increases. Each data point is an average of 20 runs, with the standard deviation being represented by the bars.The following parameters are used $K = 4, D = 30, m = 10^5, \alpha = 0.05, \rho = 0.85$.

Cost plot - Figure 4-6 - Baum-Welch, Uncoupled CPD, and Coupled CPD can all be observed to remain nearly unaffected by an increase of N. Their means fluctuate, and Baum-Welch and Uncoupled CPD have large standard deviations. When observing the box plot, Figure C-4, the same story can be seen: N has minimal to no effect on the HMM cost. It would be preferable to observe the performance of larger N, in the $10^2 - 10^4$ range, but due to time constraints in this work, that is not available.

Time plot - Figure 4-6 - The time complexity of Baum-Welch looks to be constant, but

when zoomed in it can be observed that there is a slight increase in calculation time as N increases. This is because m is relatively large for Baum-Welch, causing the forward-backward to have a large influence on the time cost making the influence of N very small in comparison. Note that the mean of Baum-Welch for $N = 2$ and $N = 50$ differ about $50[s]$ nonetheless. Uncoupled CPD and the creation of the JPTs increase with N, but not as fast as Coupled CPD. This again can be attributed to the slow permutation matching algorithm, which grows with $\mathcal{O}(N-1)$ with regard to N. From Figure C-4, one can note that for the $N = 35, 50$, the non-outlier time cost varies significantly, and that Coupled CPD is expected to surpass Baum-Welch and Uncoupled CPD in calculation time at round $N = 100$ to $N = 150$ when looking at the median.

Summary - Whilst Coupled CPD still outperforms the other methods, it can be observed to increase in calculation time faster than the other methods when N increases. Therefore, it is expected to be less time efficient than Baum-Welch and Uncoupled CPD for $N > 130$.

Number of samples m

An increase in samples makes the JPT a more accurate description of the underlying distribution. Noise cannot completely be filtered out by sampling more, but biases introduced by chance from noise or the underlying distribution will diminish as the number of samples grows. The more the underlying distribution is accurately described, the better the algorithms are capable of estimating the original HMM matrices. An increase in samples also means an increase in calculation time for the creation of the JPT. If all other variables remain the same, number of states and emissions, etc, the Uncoupled CPD or Coupled CPD should not have a calculation time increase, as it is the values of the JPT that change, but not the dimensions or rank of the tensor.

Cost plot - The cost of the Baum-Welch algorithm remains near-constant, with some variations in the cost. Baum-Welch has the benefit over CPD methods in that it performs rather well for a low number of samples. The HMM error can be seen to outperform Uncoupled CPD up to around $m = 10^5$, and Coupled CPD until $m = 10^4$.

Uncoupled CPD's HMM error converges and flattens out around $m = 10^5$, suggesting that without HMM constraints the underlying distribution in the factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ cannot be approximated more accurately after that. Note that with an increase in D, this is expected to improve. Figure C-5 also interestingly shows that Uncoupled CPD can occasionally outperform Coupled CPD when m is large, the median however can been seen to be significantly higher than that of Coupled CPD, again hinting at the instability of Uncoupled CPD, and the need for HMM constraints.

Coupled CPD can be seen to outperform both other methods in terms of cost for $m > 10^4$. The error converges faster and becomes nearly an order of magnitude smaller than Baum-Welch's error as m increases. The standard deviation of the error also diminishes as m increases. This may not appear to be the case as the error bars are visually large around $m = 10^5$; however, this is due to the plot being in log scale. The standard deviation becomes smaller closer to $m = 10^7$ since more samples create a more accurate description of the true underlying distribution, removing ambiguity due to small samples. Figure C-5 supports these statements but also shows the flattening of the HMM error convergence for large m. This

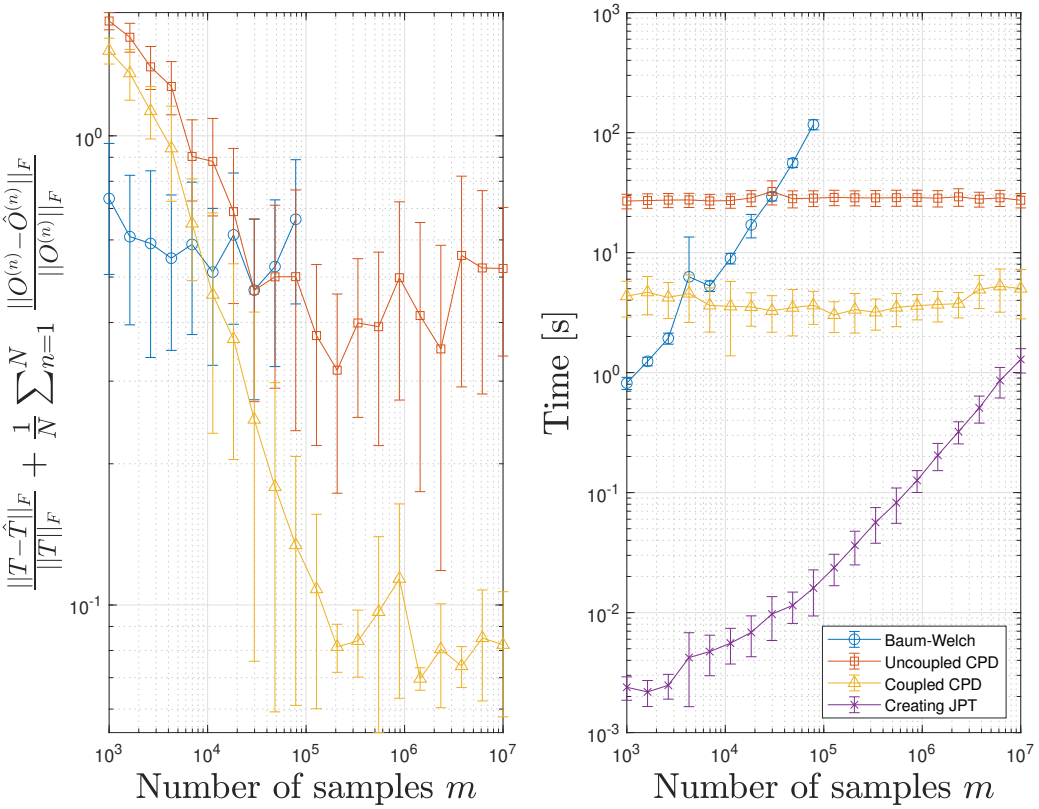


Figure 4-7: A plot of the HMM recovery error and the associated time cost, over the number of samples m for HMM learning and JPT creation. Each data-point in the plot along m is the mean of 20 runs, and the error bars on each data-point show the standard deviation of those runs. Note that each run has a random true underlying multivariate HMM and is randomly initialized; if a prior were given, the standard deviation for these methods would be smaller. The plot uses the following parameters $K = 4, D = 30, N = 5, \alpha = 0.1, \rho = 0.85$.

is consistent with the tensor convergence plot in Figure 3-2. This suggests that with more accurate approximations of the true underlying JPT, Coupled CPD could perform better.

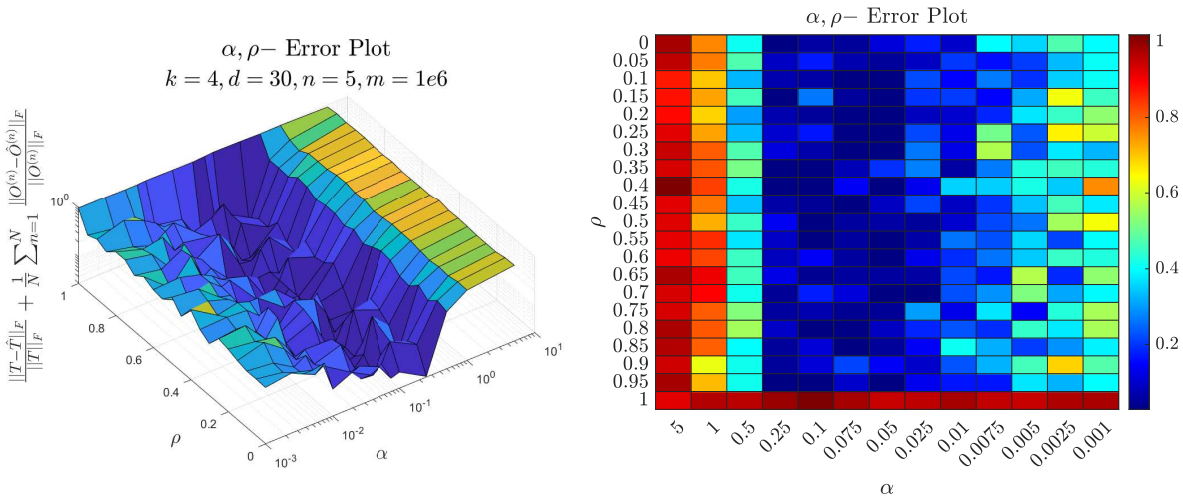
Time plot - The number of samples has a large influence on the calculation time for creating JPTs, and for Baum-Welch. The JPT creation through an indicator function can be reasoned to be linearly dependent on the number of samples in terms of calculation time. Baum-Welch employs a forward-backward pass for each iteration, which increases calculation time as the number of samples increases. Note that for Baum-Welch, the increased calculation time, but constant error, suggests that increasing the number of samples for this method is obsolete or even has a negative effect.

Uncoupled CPD and Coupled CPD remain nearly constant in calculation time as the number of samples increases. This is because the number of samples only influences the accuracy of the values inside the JPT, it does not change anything about the dimensions of the tensor $\mathbb{R}^{D\times D\times D}$, the number of tensors to consider N, or the underlying rank K. Coupled CPD can be observed to slightly outperform Uncoupled CPD in terms of calculation time. This is due

to the faster convergence. The iterations of Coupled CPD individually take longer than the ones from Uncoupled CPD due to extra overhead from the removal of permutation ambiguity.

4-1-3 Performance over α and ρ

The variables α and ρ are hyperparameters of the Coupled CPD algorithm. $\alpha > 0$ dictates how much the update equations rely on the ALS part or the HMM common $\mathbf{T}$ part. $\rho \in [0, 1]$ dictates how much the updated transition matrix $\mathbf{T}$ relies on the previous transition matrix $\mathbf{T}_{prev}$ or on the average of the found transition matrices from all N coupled subproblems. ρ is introduces as the transition matrix $\mathbf{T}$ had a tendency to oscillate in convergence.



(a) Surface plot **(b)** Heatmap

Figure 4-8: Performance of the Coupled algorithm over α, ρ for $K = 4, D = 30, N = 5, m = 10^5$. This cost manifold is sampled from 21×13 points, which are all averaged over 10 runs.

In Figure 4-8, the HMM cost over different values of α and ρ can be observed. The surface plot on the left and the heatmap on the right contain the same values. One can observe that α plays an important role in the HMM error value of the Coupled CPD algorithm. ρ plays less of a dramatic role, but can still be observed to influence the cost. When looking at the bottom row of the heatmap or the far side of the surface plot, one can notice that that area has a bad HMM error. This is because $\rho = 1$ means $\mathbf{T} = \mathbf{T}_{prev}$ in the update rules, which keeps $\mathbf{T}$ equal to its initialization. The local minima of this HMM error manifold lies at $\alpha = 0.1, \rho = 0.35$ and equal 0.0287.

4-1-4 Tensor convergence

As noted, the reconstruction error between the tensor $\underline{\mathbf{P}}^{(n)}$ and a tensor created with the factor matrices $[[\mathbf{A}^{(n)}, \mathbf{B}^{(n)}, \mathbf{C}^{(n)}]]$, is not telling of the HMM reconstruction error. That is to say, the reconstruction error between tensor and estimate can be really small, but could lack the probabilistic representation in the factor matrices, needed to construct the HMM matrices.

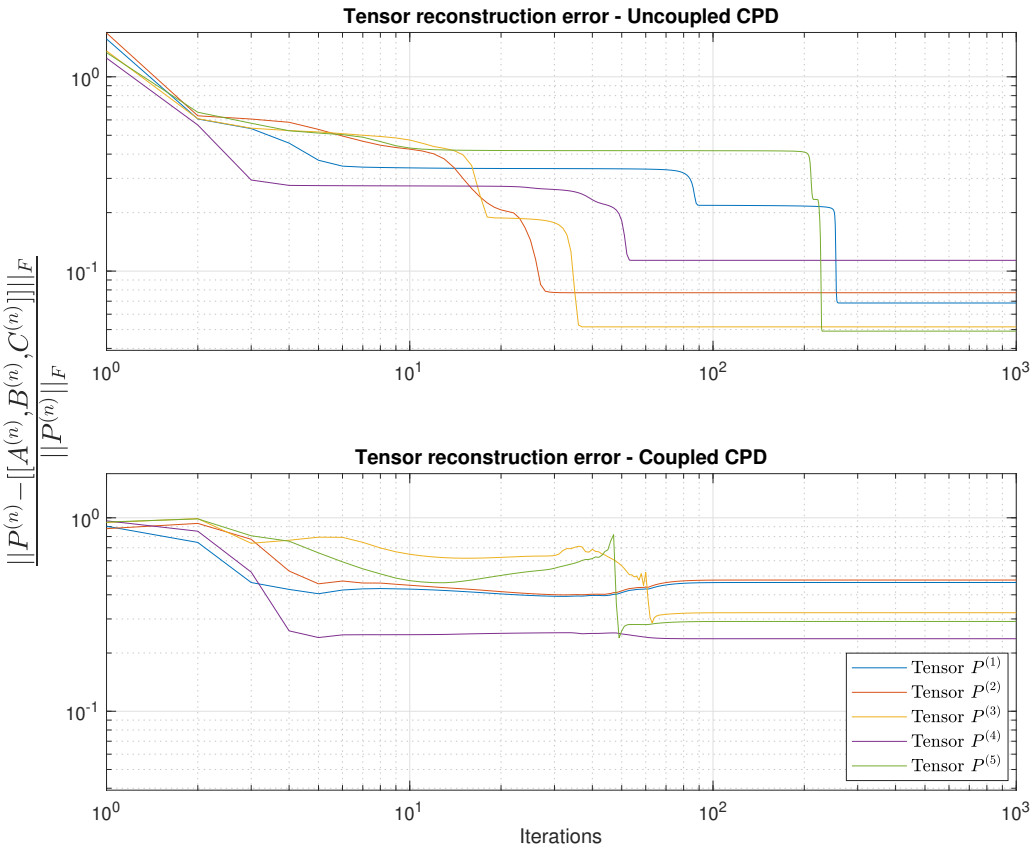


Figure 4-9: A plot of the convergence of the relative error of the JPTs that were input into Uncoupled CPD and Coupled CPD. Here $K = 4, D = 30, N = 5, m = 10^6$. The HMM cost for this particular run in the plot is 0.3940 for Uncoupled CPD and 0.0099 for Coupled CPD. The maximum iterations of both algorithms are set to 1000, and both methods are initialized with the same factor matrices.

This can be observed in Figure 4-9. Even though for Uncoupled CPD the reconstruction error of the tensors over time is clearly smaller than that of Coupled CPD, because Uncoupled CPD does not make use of HMM constraints, the HMM error (Equation (3-13)) is larger than that of Coupled CPD. In this case, where both methods have had 1000 iterations and are initialized using the same factor matrices $\mathbf{A}^{(n)}, \mathbf{B}^{(n)}, \mathbf{C}^{(n)} \ \forall n \in [1, N]$, the HMM error for Uncoupled CPD is 0.3940, and that of Coupled CPD is 0.0099.

4-1-5 Total performance

For the total performance, 100 runs are completed using parameters that generally work well for all methods and are computationally friendly, namely $K = 4, D = 30, N = 5, m = 10^5$ and for Coupled CPD $\alpha = 0.05, \rho = 0.85$. The mean and standard deviation of these runs are given in Table 4-1 for the error in the transition matrix $\mathbf{T}$, the emission matrices $\mathbf{O}^{(n)}$, the combined cost $C(\mathbf{T}, \mathbf{O}^{(n)}, \hat{\mathbf{T}}, \hat{\mathbf{O}}^{(n)})$, and the computation time.

Table 4-1: Performance over 100 runs. The following parameters are used $K = 4, D = 30, N = 5, m = 10^5, \alpha = 0.05, \rho = 0.85$. The values are given as mean $\pm$ std.

Method	$\frac{\|\|\mathbf{T}-\hat{\mathbf{T}}\|\|_F}{\|\|\mathbf{T}\|\|_F}$	$\frac{1}{N}\sum_{n=1}^{N}\frac{\|\|\mathbf{O}^{(n)}-\hat{\mathbf{O}}^{(n)}\|\|_F}{\|\|\mathbf{O}^{(n)}\|\|_F}$	$C(\mathbf{T},\mathbf{O}^{(n)},\hat{\mathbf{T}},\hat{\mathbf{O}}^{(n)})$	Time $[s]$
Baum-Welch	0.2260 ± 0.1743	0.2793 ± 0.2675	0.5052 ± 0.4332	138.15 ± 3.66
Uncoupled CPD	0.0425 ± 0.0838	0.1260 ± 0.1451	0.1685 ± 0.2265	12.21 ± 0.1781
Coupled CPD	0.0205 ± 0.0334	0.1070 ± 0.1193	0.1275 ± 0.1418	2.67 ± 2.89

From Table 4-1 it can be observed that generally Uncoupled CPD outperforms Baum-Welch and that Coupled CPD outperforms Uncoupled CPD. It can be noted that Uncoupled CPD and Coupled CPD for the HMM cost (individually for the Transition and Emission matrices, but also combined) have a higher standard deviation than the mean. Of course, when the standard deviation is negative, the cost cannot be negative. This suggests a skewness in the distribution of the cost for both methods that is not fully captured using just the values in the table. As such, the cost values are also plotted in a swarm chart in Figure 4-10. The computational time in Table 4-1 improves from Baum-Welch to Uncoupled CPD to Coupled CPD.

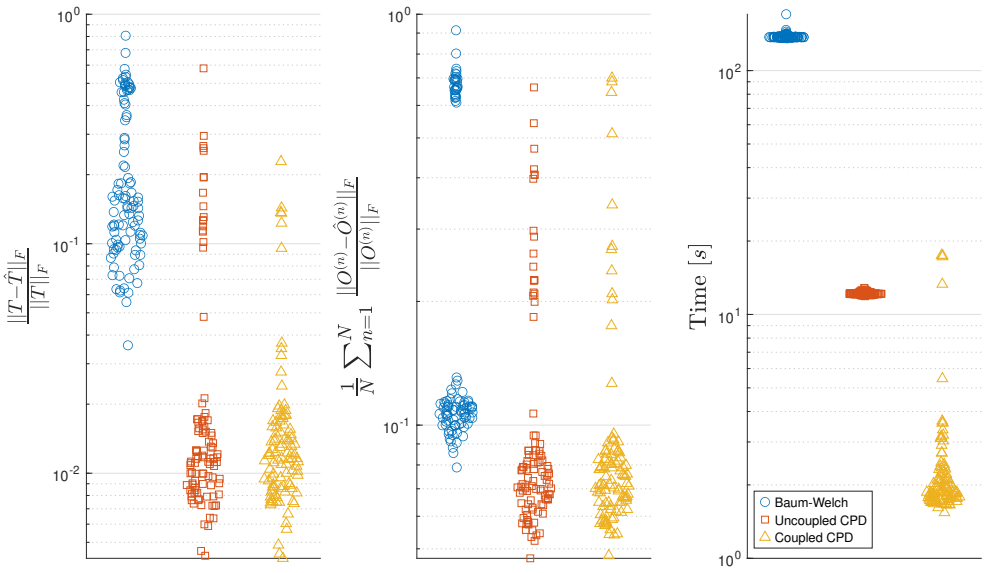


Figure 4-10: A swarm plot of Baum-Welch, Uncoupled CPD, and Coupled CPD, over 100 runs for the following parameters $K = 4, D = 30, N = 5, m = 10^5, \alpha = 0.05, \rho = 0.85$.

In Figure 4-10 for both Uncoupled CPD and Coupled CPD several outliers can be observed, particularly in the cost. These significantly pull the value of the mean seen in Table 4-1 upward, explaining the relatively high standard deviation. It can be noted that these outliers are generally still on par with the cost of Baum-Welch.

4-2 Continuous Synthetic Data

Previously, data was generated to fit in pre-defined bins using a discrete Hidden Markov Model (dHMM) as a ground truth. In this section, a continuous HMM (cHMM) is used as the underlying ground truth and is then estimated using a dHMM. This means that in each timestep, a probability density function (PDF - continuous distribution) is used to generate an emission from the state. This has a benefit over using dHMMs as ground truth, as now noise can be added to the data sequence, as can be observed in Figure 4-11.

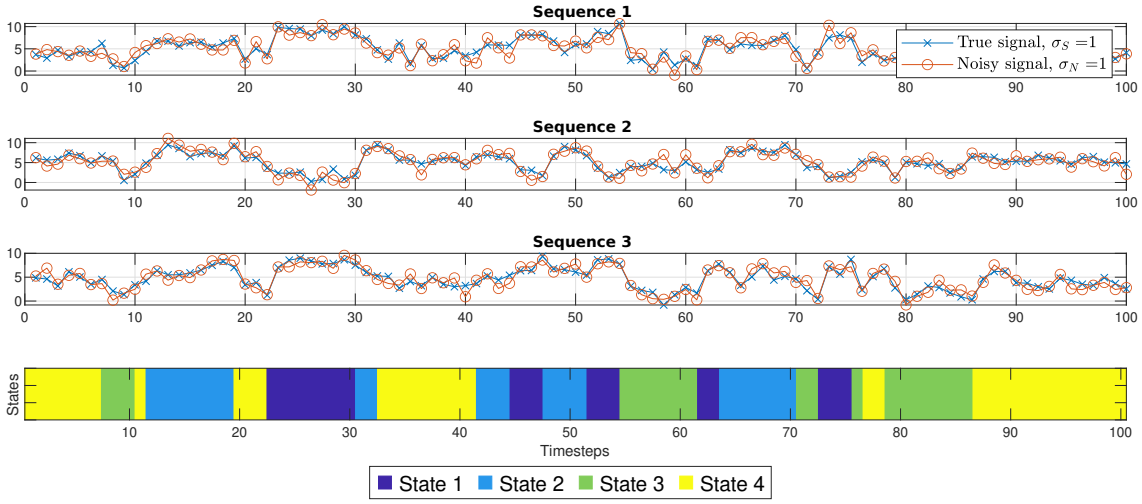


Figure 4-11: An example of continuous data from a cHMM for $K = 4, N = 3$, and $m = 100$. The true signal, sampled from the cHMM, can be observed, and that signal with added noise. Also, the underlying state evolution can be seen.

In this section, dHMMs are still used to estimate the ground truth cHMMs, which means there will always be some extra error. However, real data is generally continuous (digital but with a very high resolution), and dHMMs are generally used to fit this data. To be able to speak on the performance of the dHMM estimate obtained through this method, the ground truth cHMM needs to be discretized, as it is impossible to compare a PDF with a probability mass function (PMF). Because of this, the PDF of the cHMM will be evaluated at the bin centers from the estimated PMF from Coupled Canonical Polyadic Decomposition (CPD). This means the ground truth PDF (cHMM) will be sampled at the same position as the PMF (dHMM) of the estimated HMM, creating a ground truth dHMM which can then be compared to the estimated dHMM to quantify performance. An example of this can be observed in Figure 4-12, where the true PDF per state per data sequence can be observed, as well as where that PDF is evaluated (circles), which is at the bin centers of the estimated PMF, whose peaks are shown with crosses. For this example, the following parameters are used: $K = 4, D = 30, N = 3, m = 10^6$.

As mentioned in Section 3-1-2 and Section 3-3-5, the impact of noise on the Coupled CPD algorithm's ability to learn HMMs can be controlled by the ratio between the variance of the signal σ_S and the variance of the noise σ_N. In the examples used in cHMMs in this section, $\sigma_S = 1$. Figure 4-13 shows the effect of the variance of the noise on the HMM recovery error.

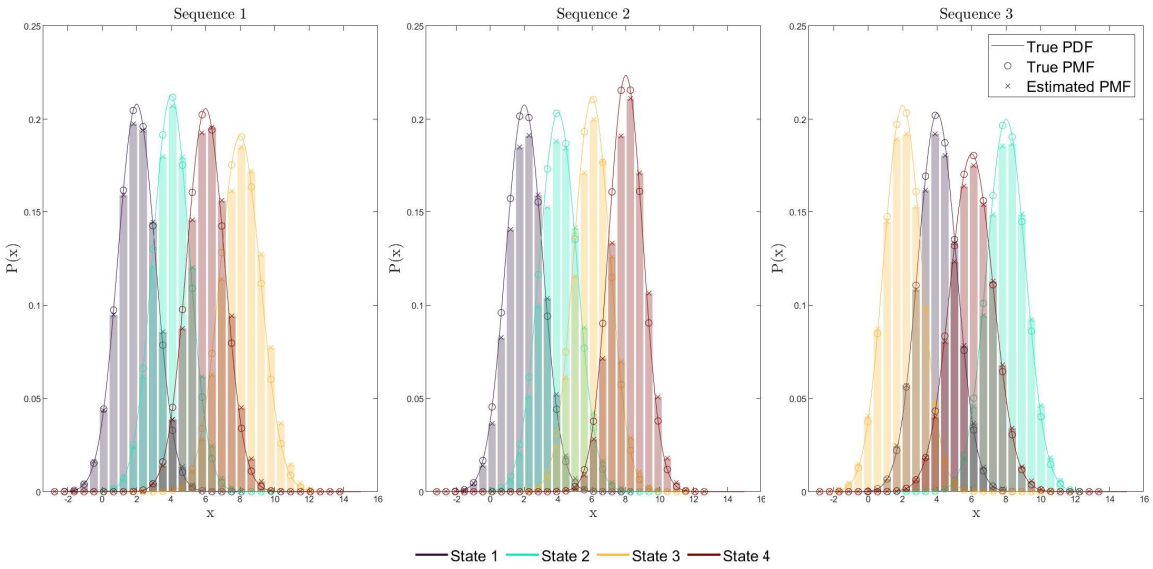


Figure 4-12: This plot shows the true underlying PDF, and a PMF which estimates this PDF through Coupled CPD. To be able to compare the two, the PDF (true PDF) is discretized into a PMF (true PMF) at the bin centers of the estimated PMF.

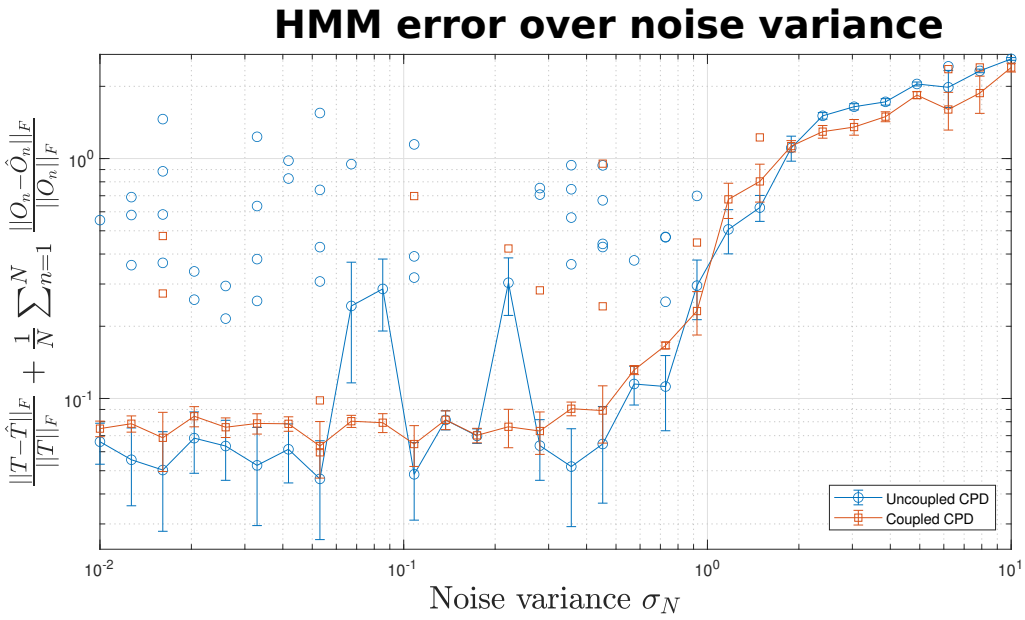


Figure 4-13: A plot of the effect of noise in the data sequences on the performance of approximating a cHMM with a dHMM. Here $\sigma_S = 1$ and σ_N varies over the x-axis. Note that this plot is averaged over 10 runs, each dot attached to a line is the mean of all the non-outliers, and the bars show the standard deviation for those non-outliers. The outliers are scattered and are not attached to a line. Note that the Uncoupled method has a lot of outliers due to the failure of the runs, and the Coupled method has fewer outliers. The plot uses the following parameters: $K = 4, D = 30, N = 5, m = 10^6$.

Figure 4-13 shows that when the noise variance σ_N is small compared to the signal's noise variance σ_S, the error stays generally low. It can also be observed that when $\sigma_N > \frac{1}{3}\sigma_S$, the error starts to grow significantly.

4-3 Real Data - Sleep stages from polysomnography

To test the Coupled Canonical Polyadic Decomposition (CPD) method in a real-life application, the Sleep Physionet dataset is used [10], which is a polysomnography dataset. An image of the sensors used in a polysomnography can be seen in Figure 4-14. This dataset is expected to operate similarly to a Hidden Markov Model (HMM) in the sense that one sleep stage is active, which equates to the states. The states are expected to be the different sleep stages, which are stage W (wakefulness), sleep stages N1 (light sleep), N2, N3 (deep sleep), and stage R (Rapid Eye Movement (REM) sleep) [28]. The effect of these states can be observed in the measurements, of which three are used in this experiment, namely two Electroencephalography (EEG) channels (EEGFpz_Cz and EEGPz_Oz) and one Electrooculography (EOG) channel (EOGHorizontal). These equate in HMM terms to the observations, emissions, or data sequences. The EEG channels record electrical activity along the scalp above the cerebral cortex. The EOG channel monitors horizontal eye movements, and REM sleep is known to have rapid eye movement [28].

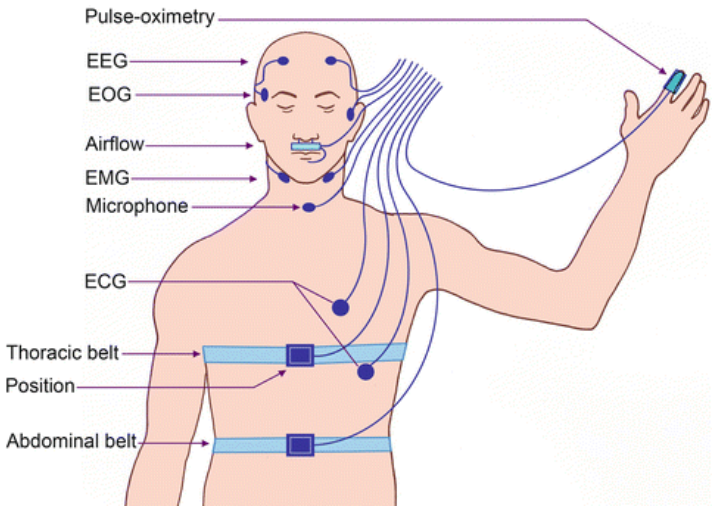


Figure 4-14: An image of all sensors in a polysomnography [28]. For this report, only the EEG and EOG channels are used as they have higher sampling rates than the other channels.

The Sleep Physionet dataset provides a range of measurements such as EEG, EOG, chin Electromyography (EMG), event markers, and some also contain respiration and body temperature. The EEG and EOG channels are sampled at 100 Hz and the others at 10 Hz. HMMs do not allow for mixed sampling rates, so only the EEG and EOG channels will be used in this experiment, see Figure 4-15. The dataset also comes with the sleep stages, which are manually annotated by well-trained technicians according to the 1968 Rechtschaffen and Kales manual [29].

Data pre-processing - This experiment closely follows [30], and as such, the EEG and EOG channels will be separated into the following power-bands: Delta δ ($0.5-4.5$ Hz), Theta

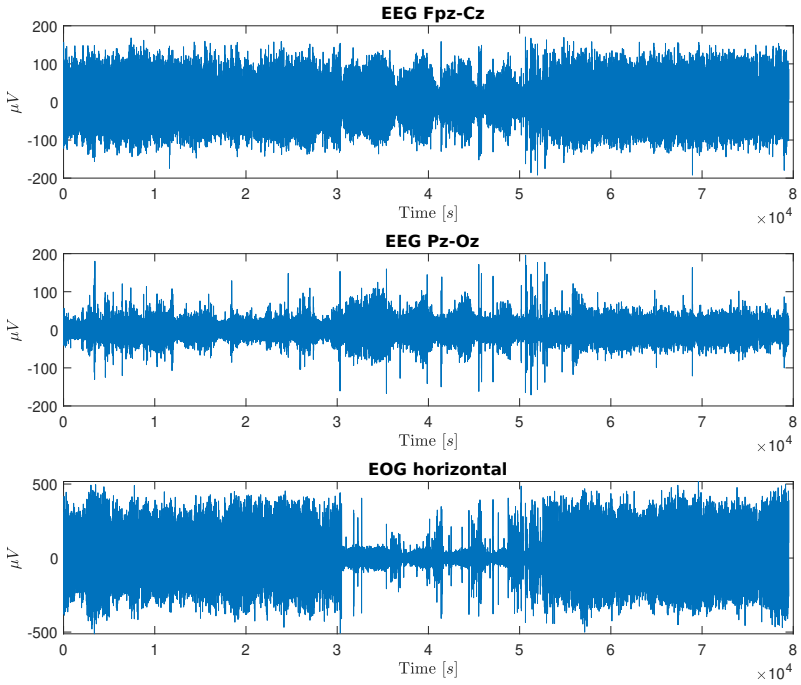


Figure 4-15: Two channels of EEG data and one channel of EOG data from a single patient 001 for a single night from the Sleep Physionet dataset [10].

θ $(4.5 - 8.5$ Hz), Alpha α $(8.5 - 11.5$ Hz), Sigma σ $(11.5 - 15.5$ Hz), and Beta β $(15.5 - 30$ Hz). These now 15 obtained signals display the local power (dB) that each of the bands contains per signal locally. These power-band signals are obtained by applying a 4th-order bandpass filter on each of the EEG and EOG channels using the aforementioned power-band frequencies. Then these filtered signals are fed into a spectrogram using a window size of $2[s]$, and an overlap of $1[s]$, and the log is taken to remove large spikes and get the data in dB range. Finally, a moving average filter is applied with a window size of 2 samples, just to filter out the most radical of noise. This results in 5 power-band signals for each of the original 3 channels, and can be seen in Figure 4-16.

Applying Coupled CPD using these 15 powerband data sequences requires the selection of a few parameters. The number of sleep stages, which represent the HMM states, is known to be W, N1, N2, N3, and REM, resulting in $K = 5$. The power bands serve as data sequences, making $N = 5$. $D = 30$ has previously been observed to work well with regard to HMM cost and computation time, and is set as such. The same is true for the hyperparameters $\alpha = 0.05$ and $\rho = 0.85$. Lastly, after the spectrogram, the number of samples is $m = 74900$, which, according to Figure 4-7, is not preferable; however, Coupled CPD should still be able to perform decently. This results in the estimates $\hat{\mathbf{T}}$ and $\hat{\mathbf{O}}^{(n)}$, which can be observed in Figure C-6b and Figure C-6d respectively.

Applying a multivariate Viterbi algorithm results in an estimation of the Viterbi path, an estimate of the underlying states or active sleep stages at each timestep. It does so by taking in the estimates $\hat{\mathbf{T}}$ and $\hat{\mathbf{O}}^{(n)}$, and the binned powerband data sequences. From

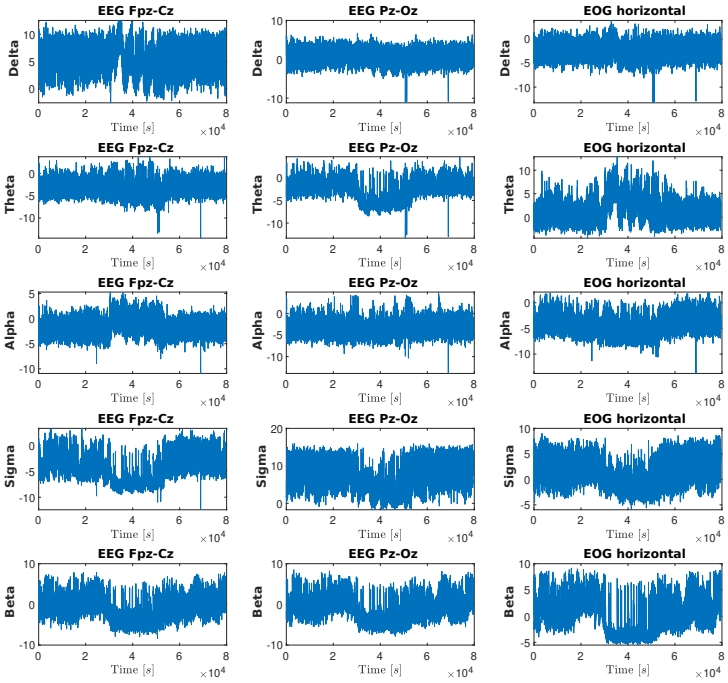


Figure 4-16: A plot of all the power bands from the original channels.

Figure 4-17, one can notice that large parts of the state sequences match. However, during sleep, the estimated sleep stages can be observed to be incorrect. Often in the estimate, N3 gets confused with N4. Also, some 'noise' can be observed all around, but is particularly visible in the wakeful stage. The estimate has a 91.2% match/accuracy with the annotated sleep stages, but this is largely due to the wakeful stage. These facts combined make it so this method can be used to estimate the difference between wakefulness and sleep, but it cannot accurately distinguish the sleep stages themselves.

Post process analysis - As the annotated sleep stages are known (in this case), as well as the discretized power-band data sequences, one can obtain the HMM matrices that most accurately fit the data and states (Figure C-6a and Figure C-6c). Here, it can be noted that in the emission matrix, in the Beta band of the EOG channel, for example, many of the distributions overlap. This is precisely what Coupled CPD struggles with, as noted in Section 3-3-2 and Section 4-1-1. Because many of these distributions are nearly identical, they cannot be learned properly by this method, and also lose meaning from an HMM perspective as independent states. In the estimates of the emission matrices through Coupled CPD (Figure C-6d), it can be noted that the same values are covered, but that Coupled CPD tries to separate them into 5 distributions for all of the states. Coupled CPD is still capable estimating the sleep data somewhat accurately, but the emissions found that data represent $P(y_t|x_t)$, and the emissions found through Coupled CPD do not. Figure C-6c shows that Coupled CPD is not fit for this particular dataset. Perhaps, through some other pre-processing steps, this could be improved.

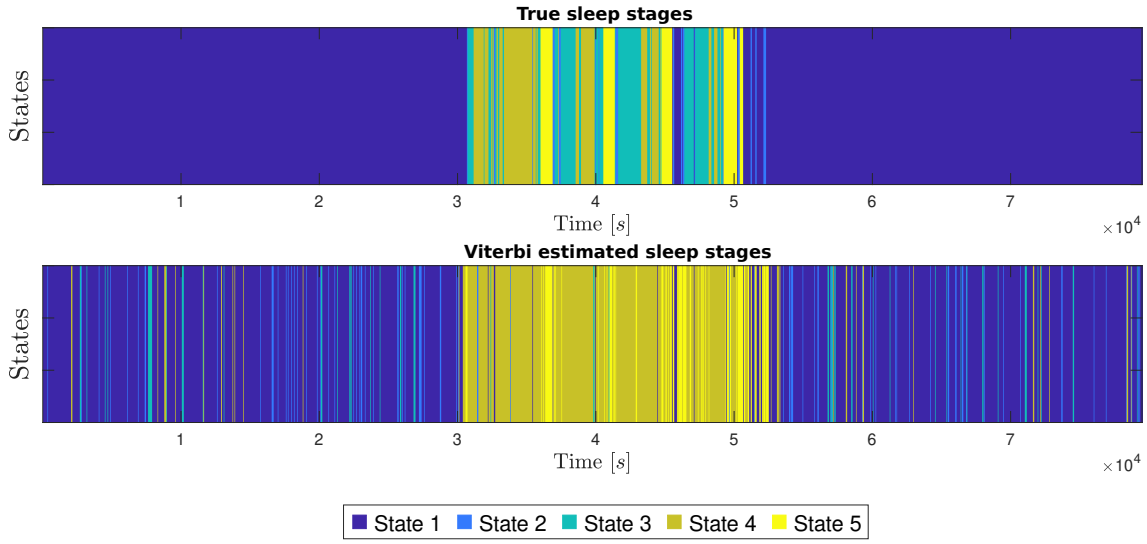


Figure 4-17: A show of the annotated sleep stages by experts, and the estimate of the sleep stages through Coupled CPD and a multivariate Viterbi algorithm. Here State 1 = W, State 2 = N1, State 3 = N2, State 4 = N3, and State 5 = REM.

4-4 Summary

This chapter has shown how the novel Coupled CPD method performs over a range of circumstances. Variations in the underlying HMMs were explored, as well as varying HMM parameters such as the number of states K, the number of emissions D, the number of data sequences N, and the number of samples per sequence m. Also, the hyperparameters used in Coupled CPD were varied to observe its response. Continuous data was explored with the addition of noise, to observe the impact of noise on the learning capabilities of Coupled CPD. Finally, real polysomnography data was used to explore a real-world application. Here Coupled CPD was used in combination with a multivariate Viterbi algorithm and some preprocessing steps to uncover the underlying sleep stages from EEG and EOG data, which were then compared to sleep stages annotated by experts.

Chapter 5

Conclusion and Future Work

In Chapter 2 theoretical background as well as related literature was discussed. It explained the description of Hidden Markov Models (HMMs) that are used in this report, as well as the concept of Joint Probability Tensors (JPTs) and Canonical Polyadic decomposition (CPD) for their deconstruction.

Then Chapter 3 explained the full pipeline of univariate HMM learning through CPD from data, and showed the instabilities it suffers from. It also offered a solution for the instabilities by coupling all problems together by realizing multivariate HMMs use the same underlying states at each timestep, and thus share a single transition matrix $\mathbf{T}$ and initial distribution $\boldsymbol{\pi}$. The chapter also discusses some considerations, metrics, and relevant functions.

Chapter 4 showed the performance of the proposed method, Coupled CPD, and compared it with the performance of the industry standard Baum-Welch and its predecessor Uncoupled CPD. It did so over a wide range of variables and parameters, such as the HMM variables K, D, N, m, for different variations in the 'to-be-learned' HMM matrices, and over the hyper-parameter of Coupled CPD itself. Also, continuous data (non-binned) with added noise is discussed, as well as a practical experiment, where Coupled CPD is applied to real polysomnography data, from the Sleep Physionet dataset [10]. This experiment showed that sleep stages can be reasonably accurately described (91.2% accuracy), but that the practical meaning of the emission probabilities is questionable, and requires extra research to be accurate.

Overall Coupled CPD is an improvement over Baum-Welch and Uncoupled CPD, with a few exceptions such as a very low number of samples m or a high number of states K. In general, all methods performed as expected concerning computation time. Going back to the main research question of this thesis:

CPD methods for HMM learning can, indeed, be improved to accommodate multivariate data sequences while outperforming the industry-standard, Baum-Welch algorithm, in both computational efficiency and estimation accuracy.

Also throughout the chapters, all sub-questions have been answered. Despite the initial success of this method and the ideas behind it, it is still in its early stages of infancy and has many areas of improvement still ahead, as discussed in the section below.

5-1 Future work

Some general suggestions that directly extend from this work are to average the performance test of K, D, N, m over more runs. Currently, they are averaged over 20 runs, but to get a more accurate description the number of runs should increase to 100-1000, although this requires significant run times. Also, the behavior of Coupled CPD with respect to the number of emissions D should be explored beyond the current $D = 100$ and the number of sequences beyond $N = 50$.

Even though Coupled CPD improved in stability compared to Uncoupled CPD, some outliers can still be present. To improve stability even further several gaps or ideas could be explored, such as uncovering why the sampled JPT does not keep converging to the calculated JPT even though the number of samples increases; or one could apply a smoothing function, or the opposite thereof, to the tensor to see if that would improve general stability.

Also, Coupled CPD could be made selective with respect to the different data sequences N. That is to say, currently, the transition matrix $\mathbf{T}$ is averaged over all data sequences, but this averaging could be weighted, based on either the correlation in the respective emission matrices or the error between the JPT of the data sequence and the reconstruction thereof through the factor matrices. By having the data sequences and thus JPTs that result in less correlation or a smaller reconstruction error have more effect on the convergence of $\mathbf{T}$ and therefore indirectly the factor matrices, the stability of Coupled CPD might improve even further.

Improving calculation time can mainly be done by swapping out the algorithm for finding matching permutations for the transition matrix, as this is by far the slowest part of Coupled CPD. A suggestion might be heuristic algorithms which are generally fast.

Finally, some literature suggests a continuous extension of tensor decompositions, that either approximate the tensor using a series of continuous functions or linearly interpolates the possible values in between existing entries in the tensor. If, for example, Gaussians would be used as basis functions, this method could perhaps be used to find continuous HMMs through a very similar method as is presented in this report.

Appendix A

Algorithms

A-1 Transition Matrix Permutation Fitting

Algorithm 2 provides pseudo-code for a transition matrix permutation fit algorithm. It intakes two transition matrices and tries to match their permutation such that matrix B is matched to matrix A. It returns a permutation matrix that provides this permutation to the transition matrix. Note that a very similar algorithm can be constructed for the emission matrices, in which case a consensus must be reached as to what permutation is taken.

Algorithm 2: Find Best Permutation to Align Two Matrices

Input : Matrix $\mathbf{A} \in \mathbb{R}^{K \times K}$, Matrix $\mathbf{B} \in \mathbb{R}^{K \times K}$
Output: Permutation matrix $\Pi_{\text{best}} \in \mathbb{R}^{K \times K}$

Initialization
  $d_A \leftarrow \text{diag}(\mathbf{A})$, $d_B \leftarrow \text{diag}(\mathbf{B})$;
  perm $\leftarrow$ empty list of length K;
Fast Diagonal Matching
  for $n = 1$ *to* K **do**
    Index $j \leftarrow \min(d_A - d_B(n))$;
    $\text{perm}(n) = j$;
  if *all entries of perm are unique* **then**
    $\Pi_{best} \leftarrow \text{permute}(I_k, \text{perm})$;
    return Π_{best} ;
Full Permutation Search
  $\mathcal{P} \leftarrow$ all $K!$ permutations of $\{1, \ldots, K\}$;
  $\text{err}_{\min} \leftarrow \infty$, $\Pi_{\text{best}} \leftarrow I_k$;
  foreach $p \in \mathcal{P}$ **do**
    Construct permutation matrix Π_p from p;
    $\mathbf{B}_p \leftarrow \Pi_p \mathbf{B} \Pi_p^T$;
    $\text{err} \leftarrow ||\mathbf{A} - \mathbf{B}_p||_F$;
    if $err < err_{\min}$ **then**
      $\text{err}_{\min} \leftarrow \text{err}$;
      $\Pi_{\text{best}} \leftarrow \Pi_p$;
  return Π_{best}

Appendix B

Mathematical Derivations

This appendix provides some math to support the minimization problems found within this report. Here follow some matrix rules from [31] which are commonly used:

$$\begin{aligned} \partial(X+Y) &= \partial X + \partial Y \\ \partial(\mathrm{Tr}(X)) &= \mathrm{Tr}(\partial X) \\ \frac{\partial}{\partial X}\mathrm{Tr}(XA) &= A^T \\ \frac{\partial}{\partial X}\mathrm{Tr}(X^TA) &= A \\ \frac{\partial}{\partial X}\mathrm{Tr}(AX^T) &= A \end{aligned} \tag{B-1}$$

B-1 ALS update equation derivation

In this report, we want to learn a 3-way tensor with Canonical Polyadic Decomposition using Alternating Least Squares. This tensor is factored into 3 learnable matrices $\underline{\mathbf{P}} = [[\mathbf{A}, \mathbf{B}, \mathbf{C}]]$. This results in the following minimization problem, with some additional constraints which are implemented due to the matrices having a probabilistic meaning:

$$\underset{\mathbf{A},\mathbf{B},\mathbf{C}}{\arg\min} \quad ||\underline{\mathbf{P}} - [[\mathbf{A}, \mathbf{B}, \mathbf{C}]]||_F^2 \tag{B-2}$$

$s.t.$

$$\begin{aligned} &\mathbf{A}, \mathbf{B}, \mathbf{C} > 0 \quad \text{(element-wise)} \\ &\sum_{k=1}^{K} \mathbf{A}(:,k) = 1, \quad \sum_{k=1}^{K} \mathbf{B}(:,k) = 1, \quad \sum_{k=1}^{K} \mathbf{C}(:,k) = 1 \end{aligned}$$

This problem can be unfolded to:

$$||\mathbf{P}_{(1)} - \mathbf{A}(\mathbf{C} \odot \mathbf{B})^T||_F^2 = ||\mathbf{P}_{(2)} - \mathbf{B}(\mathbf{C} \odot \mathbf{A})^T||_F^2 = ||\mathbf{P}_{(3)} - \mathbf{C}(\mathbf{B} \odot \mathbf{A})^T||_F^2 \tag{B-3}$$

This leads to the following minimization problems that are solved iteratively:

$$\begin{aligned}
\hat{\mathbf{A}} &\leftarrow \arg\min_{\hat{\mathbf{A}}} J_A, \quad J_A = ||\mathbf{P}_{(1)} - \hat{\mathbf{A}}(\mathbf{C} \odot \mathbf{B})^T||_F^2 \\
\hat{\mathbf{B}} &\leftarrow \arg\min_{\hat{\mathbf{B}}} J_B, \quad J_B = ||\mathbf{P}_{(2)} - \hat{\mathbf{B}}(\mathbf{C} \odot \mathbf{A})^T||_F^2 \\
\hat{\mathbf{C}} &\leftarrow \arg\min_{\hat{\mathbf{C}}} J_C, \quad J_C = ||\mathbf{P}_{(3)} - \hat{\mathbf{C}}(\mathbf{B} \odot \mathbf{A})^T||_F^2
\end{aligned} \tag{B-4}$$

Then for ALS the gradient of the sub-cost functions is set to zero, which represents a cost function minimum. Note that $||X||_F^2 = \text{Tr}(X^T X)$ and $\nabla_X \text{Tr}\left((A - XW)^T(A - XW)\right) = -2(A - XW)W^T$, see [31]. And from [32], we see $(A \odot B)^T(A \odot B) = A^T A \circledast B^T B$, leading to:

$$\begin{aligned}
J_A &= ||\mathbf{P}_{(1)} - \hat{\mathbf{A}}(\mathbf{C} \odot \mathbf{B})^T||_F^2 = \text{Tr}\left((\mathbf{P}_{(1)} - \hat{\mathbf{A}}(\mathbf{C} \odot \mathbf{B})^T)^T(\mathbf{P}_{(1)} - \hat{\mathbf{A}}(\mathbf{C} \odot \mathbf{B})^T)\right) \\
\nabla_A J_A &= -2(\mathbf{P}_{(1)} - \hat{\mathbf{A}}(\mathbf{C} \odot \mathbf{B})^T)(\mathbf{C} \odot \mathbf{B}) = 0 \\
\nabla_A J_A &= \mathbf{P}_{(1)}(\mathbf{C} \odot \mathbf{B}) - \hat{\mathbf{A}}\left((\mathbf{C}^T\mathbf{C}) \circledast (\mathbf{B}^T\mathbf{B})\right) = 0 \\
\hat{\mathbf{A}} &= \mathbf{P}_{(1)}(\mathbf{C} \odot \mathbf{B})\left[(\mathbf{C}^T\mathbf{C}) \circledast (\mathbf{B}^T\mathbf{B})\right]^\dagger
\end{aligned} \tag{B-5}$$

We do something similar for $\mathbf{B}$ and $\mathbf{C}$, after which we obtain the full update equations. After each update, which are multi-linear due to the other two factor matrices being fixed, the factor matrices are sure to be made stochastic using Equation (3-14):

$$\begin{aligned}
\mathbf{A} &\leftarrow \mathbf{P}_{(1)}(\mathbf{C} \odot \mathbf{B})\left[(\mathbf{C}^T\mathbf{C}) \circledast (\mathbf{B}^T\mathbf{B})\right]^\dagger \\
\mathbf{A} &\leftarrow \psi(\mathbf{A}) \\
\\
\mathbf{B} &\leftarrow \mathbf{P}_{(2)}(\mathbf{C} \odot \mathbf{A})\left[(\mathbf{C}^T\mathbf{C}) \circledast (\mathbf{A}^T\mathbf{A})\right]^\dagger \\
\mathbf{B} &\leftarrow \psi(\mathbf{B}) \\
\\
\mathbf{C} &\leftarrow \mathbf{P}_{(3)}(\mathbf{B} \odot \mathbf{A})\left[(\mathbf{B}^T\mathbf{B}) \circledast (\mathbf{A}^T\mathbf{A})\right]^\dagger \\
\mathbf{C} &\leftarrow \psi(\mathbf{C})
\end{aligned} \tag{B-6}$$

Note that $\left[(\mathbf{C}^T\mathbf{C}) \circledast (\mathbf{B}^T\mathbf{B})\right] \in \mathbb{R}^{K \times K}$ making the calculation time of its inverse faster then the pseudo-inverse for $(\mathbf{C} \odot \mathbf{B})^T \in \mathbb{R}^{K \times D^2}$.

B-2 Extended update equations proof

For Coupled CPD we have the same cost function as Uncoupled CPD, but tie the problems together using constraints for the factor matrix $\mathbf{A}$ and $\mathbf{C}$, which enforce a common transition matrix $\mathbf{T}$ and initial distribution $\boldsymbol{\pi}$:

$$\underset{\{\mathbf{A}^{(n)},\mathbf{B}^{(n)},\mathbf{C}^{(n)}\}_{n=1}^{N}}{\arg\min} \quad \sum_{n=1}^{N} ||\underline{\mathbf{P}}^{(n)} - [[\mathbf{A}^{(n)}, \mathbf{B}^{(n)}, \mathbf{C}^{(n)}]]||_F^2 \tag{B-7}$$

s.t.

$$\begin{aligned}
&\mathbf{A}^{(n)}, \mathbf{B}^{(n)}, \mathbf{C}^{(n)} > 0 \quad \text{(element-wise)} \\
&\sum_{k=1}^{K} \mathbf{A}^{(n)}(:,k) = 1, \quad \sum_{k=1}^{K} \mathbf{B}^{(n)}(:,k) = 1, \quad \sum_{k=1}^{K} \mathbf{C}^{(n)}(:,k) = 1 \\
&\mathbf{A}^{(n)} = \mathbf{B}^{(n)}\text{diag}(\boldsymbol{\pi})\mathbf{T}^T\text{diag}(\mathbf{T}\boldsymbol{\pi})^{-1} \\
&\mathbf{C}^{(n)} = \mathbf{B}^{(n)}\mathbf{T} \\
&\mathbf{T} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{B}^{(n)})^{\dagger}\mathbf{C}^{(n)} \\
&\boldsymbol{\pi}^T\mathbf{T} = \boldsymbol{\pi}^T
\end{aligned}$$

To enforce the constraints, and avoid oscillation in the factor matrices, a soft scaling factor λ is introduced. Each factor matrix now has three update steps; Multi-linear update using the other fixed factor matrices; soft scaling using λ which takes a part of the factor matrix and a part of the constraints; A projection using $\psi(\cdot)$ to make sure the columns of the factor matrices are stochastic.

$$\begin{aligned}
&\mathbf{A}^{(n)} \leftarrow \mathbf{P}_{(1)}(\mathbf{C}^{(n)} \odot \mathbf{B}^{(n)}) \left[\left((\mathbf{C}^{(n)})^T\mathbf{C}^{(n)}\right) \circledast \left((\mathbf{B}^{(n)})^T\mathbf{B}^{(n)}\right)\right]^{\dagger} \\
&\mathbf{A}^{(n)} \leftarrow \lambda\mathbf{A}^{(n)} + (1-\lambda)\cdot\mathbf{B}^{(n)}\text{diag}(\boldsymbol{\pi})\mathbf{T}^T\text{diag}(\mathbf{T}\boldsymbol{\pi})^{-1}, \\
&\mathbf{A}^{(n)} \leftarrow \psi(\mathbf{A}^{(n)}) \\
\\
&\mathbf{B}^{(n)} \leftarrow \mathbf{P}_{(2)}(\mathbf{C}^{(n)} \odot \mathbf{A}^{(n)}) \left[\left((\mathbf{C}^{(n)})^T\mathbf{C}^{(n)}\right) \circledast \left((\mathbf{A}^{(n)})^T\mathbf{A}^{(n)}\right)\right]^{\dagger} \\
&\mathbf{B}^{(n)} \leftarrow \lambda\mathbf{B}^{(n)} + (1-\lambda)\mathbf{B}^{(n)} \qquad \text{(redundant)} \\
&\mathbf{B}^{(n)} \leftarrow \psi(\mathbf{B}^{(n)}) \\
\\
&\mathbf{C}^{(n)} \leftarrow \mathbf{P}_{(3)}(\mathbf{B}^{(n)} \odot \mathbf{A}^{(n)}) \left[\left((\mathbf{B}^{(n)})^T\mathbf{B}^{(n)}\right) \circledast \left((\mathbf{A}^{(n)})^T\mathbf{A}^{(n)}\right)\right]^{\dagger} \\
&\mathbf{C}^{(n)} \leftarrow \lambda\mathbf{C}^{(n)} + (1-\lambda)\cdot\mathbf{B}^{(n)}\mathbf{T}, \\
&\mathbf{C}^{(n)} \leftarrow \psi(\mathbf{C}^{(n)})
\end{aligned} \tag{B-8}$$

Here it can also be proved that the soft scaling is in fact the same as enforcing the hard constraint. Lets take $\mathbf{C}$'s soft scaling as it is the easier example:

$$\begin{aligned}
\mathbf{C}^{(n)} &= \lambda \mathbf{C}^{(n)} + (1-\lambda) \cdot \mathbf{B}^{(n)}\mathbf{T} \\
(1-\lambda)\mathbf{C}^{(n)} &= (1-\lambda)\mathbf{B}^{(n)}\mathbf{T} \\
\mathbf{C}^{(n)} &= \frac{(1-\lambda)}{(1-\lambda)}\mathbf{B}^{(n)}\mathbf{T} \\
\mathbf{C}^{(n)} &= \mathbf{B}^{(n)}\mathbf{T}
\end{aligned}$$

Finally, because of the normalization through ψ, the total scaling is irrelevant between steps 2 and 3 of the update steps in Equation (B-8) factor matrix resulting in the following equation, that uses $\alpha = \frac{1-\lambda}{\lambda}$ as the soft-scaling variable instead, which is used in Section 3-2:

$$\begin{aligned}
\frac{\mathbf{C}^{(n)}}{\lambda} &= \mathbf{C}^{(n)} + \frac{1-\lambda}{\lambda} \cdot \mathbf{B}^{(n)}\mathbf{T} \\
&= \mathbf{C}^{(n)} + \alpha \cdot \mathbf{B}^{(n)}\mathbf{T} \\
&\underset{\psi(\mathbf{C}^{(n)})}{\Longrightarrow} \mathbf{C}^{(n)} = \mathbf{C}^{(n)} + \alpha \cdot \mathbf{B}^{(n)}\mathbf{T}
\end{aligned}$$

Appendix C

Extra Figures

This appendix provides some supplementary plots.

C-1 Performance box-plots

The performance box-plots give some extra insight into the shape of the distributions of the behavior over the HMM parameters. That is to say, they show where the median, minimum, maximum, and outliers are.

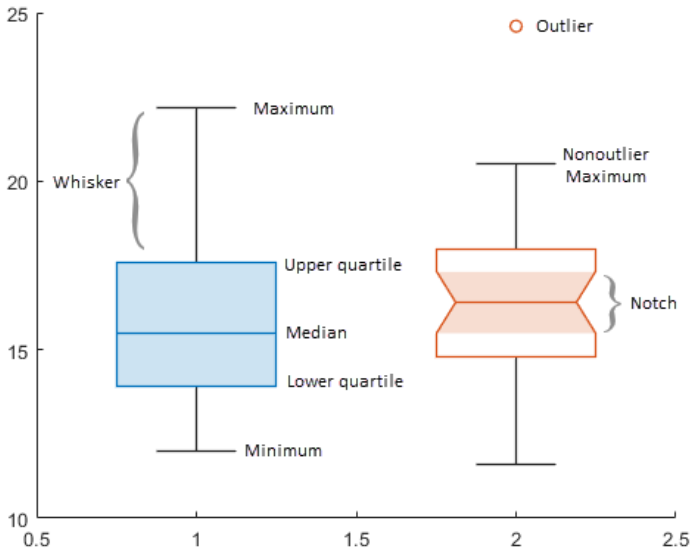


Figure C-1: A plot showing how to read box plots.

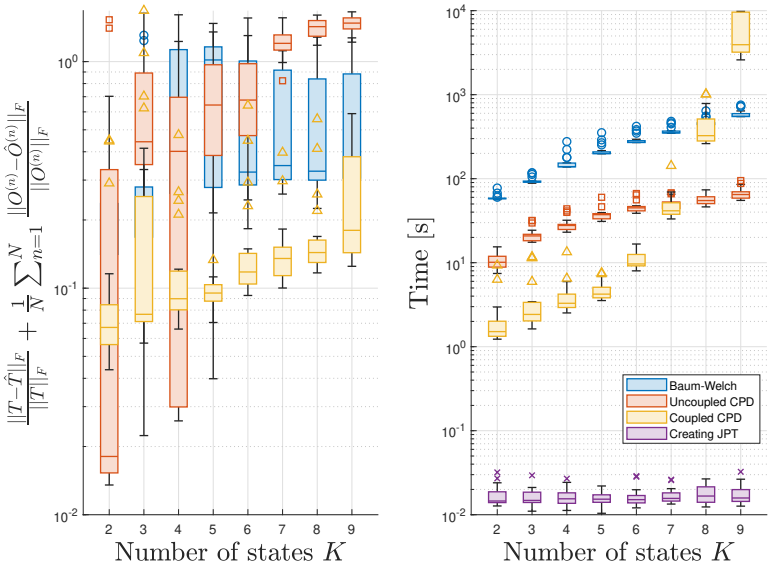


Figure C-2: A box plot of the performance of Baum-Welch, Uncoupled CPD, and Coupled CPD over the number of states. Also, the time performance is present. Each has a different marker style, Baum-Welch has circles, Uncoupled CPD has squares, Coupled CPD has triangles, and creating the JPT has crosses.

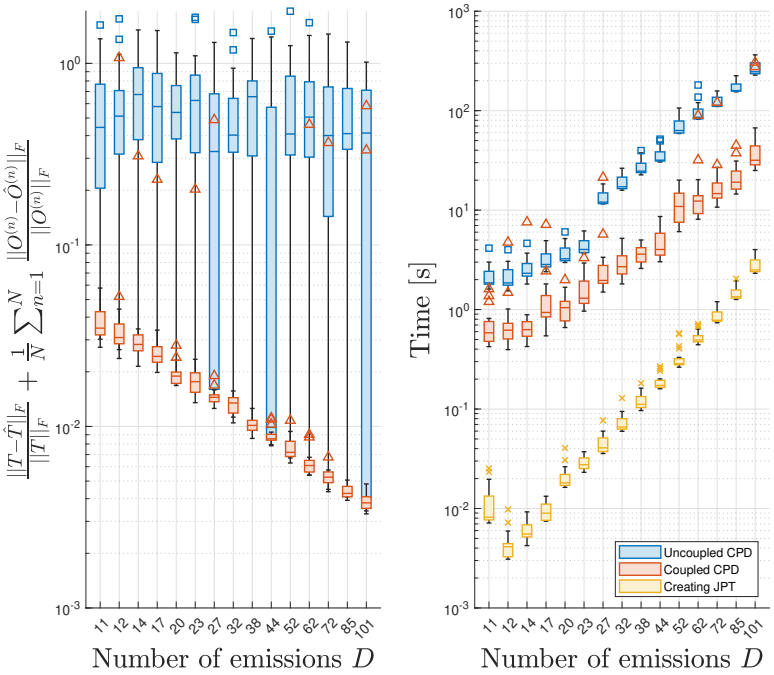


Figure C-3: A box plot of the performance of Baum-Welch, Uncoupled CPD, and Coupled CPD over the number of emissions. Also, the time performance is present. Each has a different marker style, Uncoupled CPD has squares, Coupled CPD has triangles, and creating the JPT has crosses.

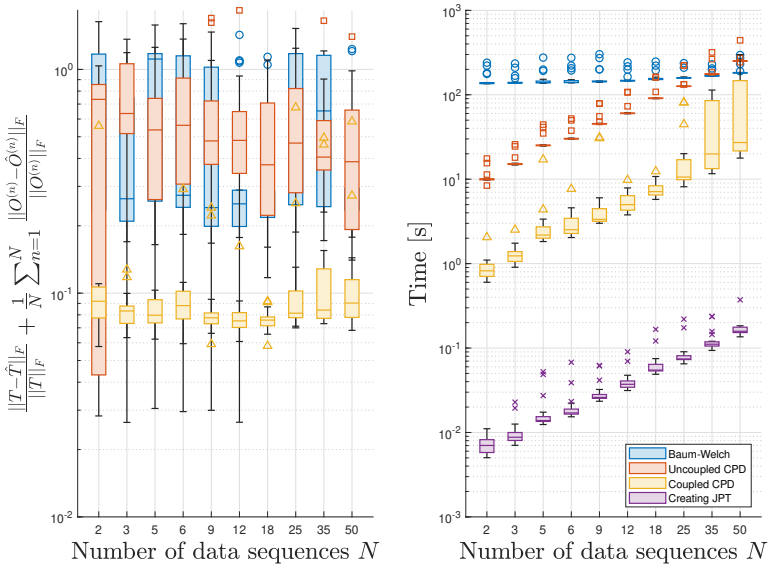


Figure C-4: A box plot of the performance of Baum-Welch, Uncoupled CPD, and Coupled CPD over the number of sequences. Also, the time performance is present. Each has a different marker style, Baum-Welch has circles, Uncoupled CPD has squares, Coupled CPD has triangles, and creating the JPT has crosses.

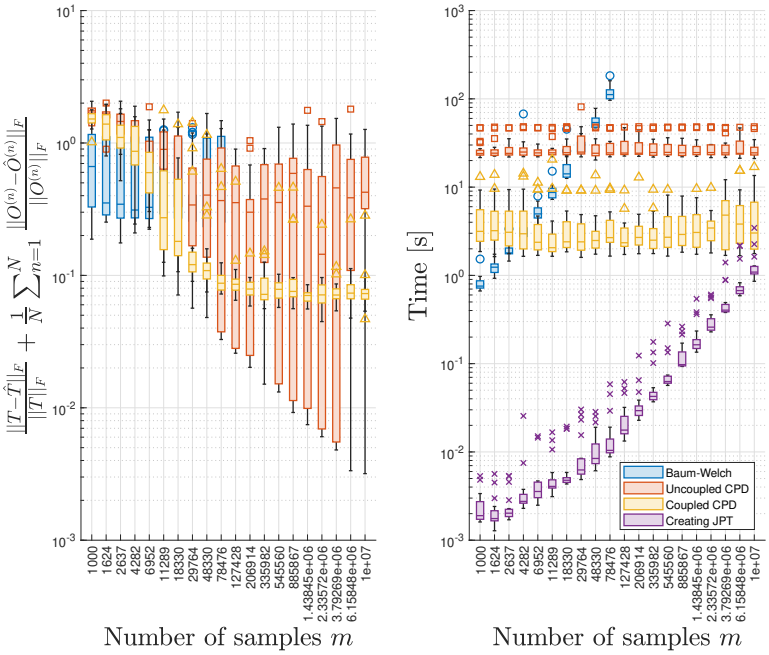
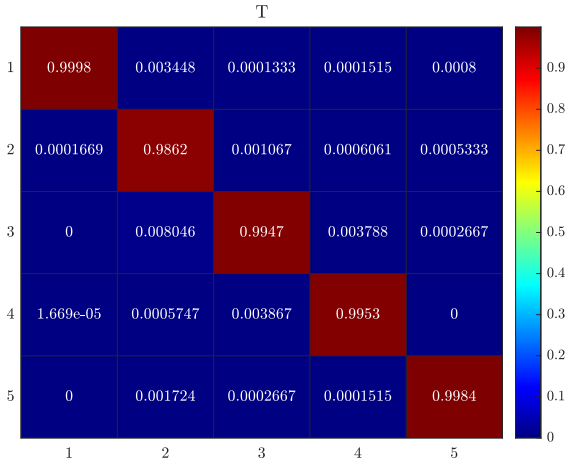
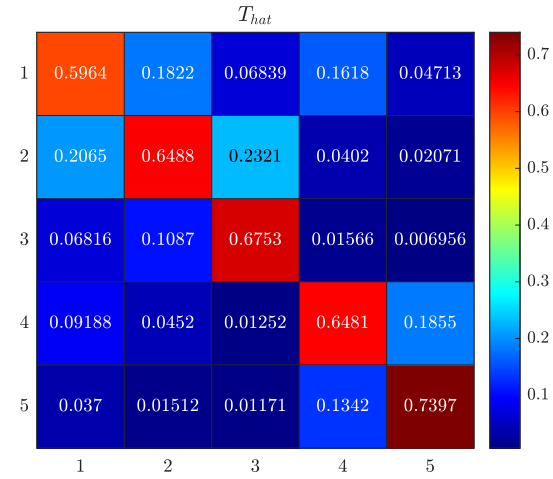


Figure C-5: A box plot of the performance of Baum-Welch, Uncoupled CPD, and Coupled CPD over the number of samples. Also, the time performance is present. Each has a different marker style, Baum-Welch has circles, Uncoupled CPD has squares, Coupled CPD has triangles, and creating the JPT has crosses.

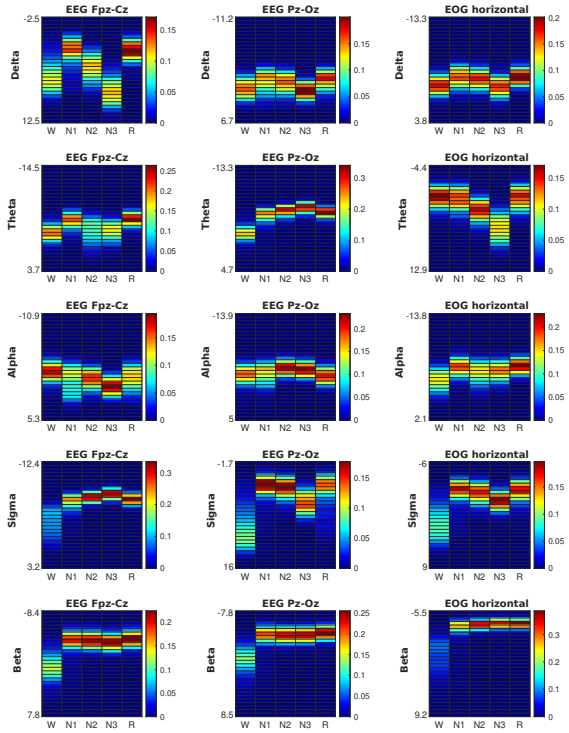
C-2 Sleep data HMM matrices



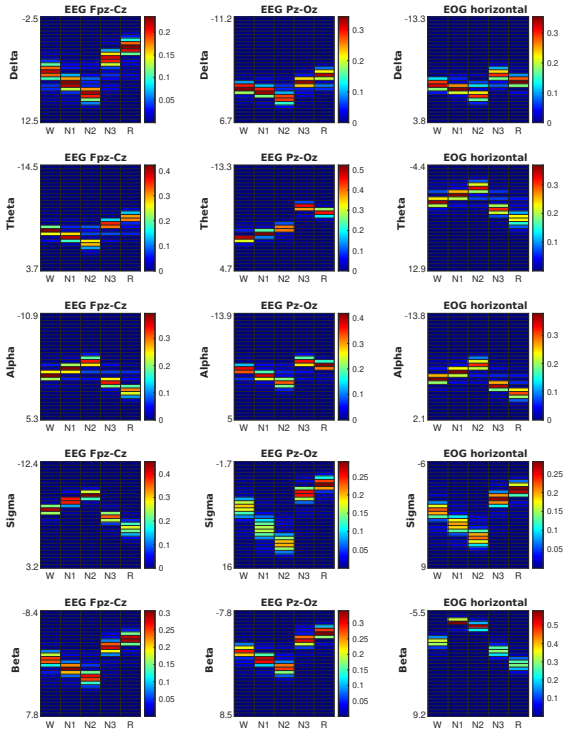
(a) Transition matrix for the sleep stages.



(b) Coupled CPD estimate of the transition matrix for sleep stages.



(c) Emission matrices for the power-band data sequences.



(d) Coupled CPD estimate of the emission matrices for each of the power-band data sequences.

Figure C-6: The transition matrix and emission matrices found through sleep stage annotations and power-band data (left), and an estimate of them through Coupled CPD (right).

The sleep data HMM matrices show the distribution of the underlying HMM to the data as obtained from the annotated states in combination with the data, and those obtained purely from data through Coupled CPD.

Bibliography

[1] B. Hunyadi, M. W. Woolrich, A. J. Quinn, D. Vidaurre, and M. D. Vos, "A dynamic system of brain networks revealed by fast transient eeg fluctuations and their fmri correlates," *NeuroImage*, vol. 185, pp. 72–82, 1 2019.

[2] A. P. Baker, M. J. Brookes, I. A. Rezek, S. M. Smith, T. Behrens, P. J. Smith, and M. Woolrich, "Fast transient networks in spontaneous human brain activity," *eLife*, vol. 2014, 3 2014.

[3] D. Vidaurre, A. J. Quinn, A. P. Baker, D. Dupret, A. Tejero-Cantero, and M. W. Woolrich, "Spectrally resolved fast transient brain states in electrophysiological data," *NeuroImage*, vol. 126, pp. 81–95, 2 2016.

[4] R. Mattila, C. Rojas, E. Moulines, V. Krishnamurthy, and B. Wahlberg, "Fast and consistent learning of hidden Markov models by incorporating non-consecutive correlations," in *Proceedings of the 37th International Conference on Machine Learning* (H. D. III and A. Singh, eds.), vol. 119 of *Proceedings of Machine Learning Research*, pp. 6785–6796, PMLR, 13–18 Jul 2020.

[5] D. Hsu, S. M. Kakade, and T. Zhang, "A spectral algorithm for learning hidden markov models," in *Journal of Computer and System Sciences*, vol. 78, pp. 1460–1480, Academic Press Inc., 2012.

[6] P. Tune, H. X. Nguyen, and M. Roughan, "Hidden markov model identifiability via tensors," in *2013 IEEE International Symposium on Information Theory*, pp. 2299–2303, 2013.

[7] K. Huang, X. Fu, and N. Sidiropoulos, "Learning hidden Markov models from pairwise co-occurrences with application to topic modeling," in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 2068–2077, PMLR, 10–15 Jul 2018.

[8] M. Cibula and R. Marik, "Tensor decomposition-based training method for high-order hidden markov models," in *Conference on Theory and Practice of Information Technologies*, 2021.

[9] M. A. Kuznetsov and I. V. Oseledets, "Tensor train spectral method for learning of hidden markov models (hmm)," *Computational Methods in Applied Mathematics*, vol. 19, pp. 93–99, 1 2019.

[10] B. Kemp, A. Zwinderman, B. Tuk, H. Kamphuisen, and J. Oberye, "Analysis of a sleep-dependent neuronal feedback loop: the slow-wave microcontinuity of the eeg," *IEEE Transactions on Biomedical Engineering*, vol. 47, no. 9, pp. 1185–1194, 2000.

[11] S. Sagitov, D. Stirzaker, and G. Grimmett, "Lecture notes on: Probability and random processes (book)," tech. rep., Chalmers University of Technology and Gothenburg University, 8 2013.

[12] J. R. Norris, "Markov chains," p. 237, Cambridge University Press, 1997.

[13] P. Baggenstoss, "A modified baum-welch algorithm for hidden markov models with multiple observation spaces," *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 4, pp. 411–416, 2001.

[14] A. Anandkumar, D. Hsu, and S. M. Kakade, "A method of moments for mixture models and hidden markov models," 3 2012.

[15] A. Anandkumar, D. Hsu, S. M. Kakade, and M. Telgarsky, "Tensor decompositions for learning latent variable models," 2014.

[16] C. Mattfeld, "Implementing spectral methods for hidden markov models with real-valued emissions," 4 2014.

[17] E. Skau, A. Hollis, S. Eidenbenz, K. Rasmussen, and B. Alexandrov, "Generating hidden markov models from process models through nonnegative tensor factorization," 4 2024.

[18] Q. Huang, R. Ge, S. Kakade, and M. Dahleh, "Minimal realization problems for hidden markov models," 11 2014.

[19] I. Domanov and L. D. Lathauwer, "Canonical polyadic decomposition of third-order tensors: Relaxed uniqueness conditions and algebraic algorithm," *Linear Algebra and Its Applications*, vol. 513, pp. 342–375, 1 2017.

[20] A. Cichocki, N. Lee, I. Oseledets, A. H. Phan, Q. Zhao, and D. P. Mandic, "Tensor networks for dimensionality reduction and large-scale optimization part 1 low-rank tensor decompositions," *Foundations and Trends in Machine Learning*, vol. 9, pp. 249–429, 2016.

[21] A. Cichocki, A.-H. Phan, Q. Zhao, N. Lee, I. V. Oseledets, M. Sugiyama, and D. Mandic, "Tensor networks for dimensionality reduction and large-scale optimizations. part 2 applications and future perspectives," 8 2017.

[22] S. Poushpas, P. Normahani, I. Kisil, B. Szubert, D. P. Mandic, and U. Jaffer, "Tensor decomposition and machine learning for the detection of arteriovenous fistula stenosis: An initial evaluation," *PLoS ONE*, vol. 18, 7 2023.

[23] K. Xie, J. Yu, and C. Lu, "A new canonical polyadic decomposition algorithm with improved stability and its applications to biomedical signal processing," *Cluster Computing*, vol. 20, pp. 1449–1455, 6 2017.

[24] O. Carr, F. Andreotti, K. Saunders, N. Palmius, G. Goodwin, and M. de Vos, "Monitoring depression in bipolar disorder using circadian measures from smartphone accelerometers," 07 2020.

[25] Q. Huang, D. Cohen, S. Komarzynski, X.-M. Li, P. Innominato, F. Lévi, and B. Finkenstädt, "Hidden markov models for monitoring circadian rhythmicity in telemetric activity data," *Journal of The Royal Society Interface*, vol. 15, p. 20170885, 02 2018.

[26] O. Carr, "Hidden markov model toolbox for matlab." https://github.com/oliver-carr/Time-Varying-Hidden-Markov-Model, 2020.

[27] V. de Silva and L.-H. Lim, "Tensor rank and the ill-posedness of the best low-rank approximation problem," 2008.

[28] S. R. Pandi-Perumal, D. W. Spence, and A. S. BaHammam, *Polysomnography: An Overview*, pp. 29–42. New York, NY: Springer New York, 2014.

[29] A. K. A Rechtschaffen, *A manual of standardized terminology, techniques and scoring systems for sleep stages of human subjects*, 10 1968.

[30] S. Chambon, M. N. Galtier, P. J. Arnal, G. Wainrib, and A. Gramfort, "A deep learning architecture for temporal sleep stage classification using multivariate and multimodal time series," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 26, no. 4, pp. 758–769, 2018.

[31] K. B. Petersen and M. S. Pedersen, "The matrix cookbook," nov 2012. Version 20121115.

[32] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.