Quantum-inspired Machine Learning with Hidden Quantum Markov Models and Tensor Networks

Siddarth Srinivasan

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington
2022

Reading Committee:
Byron Boots, Chair
Sewoong Oh
Jamie Morgenstern
Geoff Gordon

Program Authorized to Offer Degree:
Computer Science & Engineering

University of Washington

Abstract

Quantum-inspired Machine Learning with Hidden Quantum Markov Models and Tensor Networks

Siddarth Srinivasan

Chair of the Supervisory Committee:
Byron Boots
Computer Science & Engineering

The prospect of blending ideas from quantum information and machine learning has garnered interest in recent years, driven by their shared mathematical foundations in linear algebra and probability. A common way to categorize the research directions in this space is in terms of the goals (whether they are tackling classical or quantum problems) and methods (whether they rely on quantum-inspired classical computation or quantum computation). This work focuses on the potential of quantum-inspired classical machine learning approaches for solving select classical and quantum problems. In particular, we present our work on three main topics: (1) our formulation and learning algorithm for hidden quantum Markov models (HQMMs), a quantum-inspired analogue of hidden Markov models (HMMs) with greater expressiveness and without the learning challenges associated with previous proposals extending HMMs, (2) the connection between HQMMs (and similar proposals) and tensor networks, a general and tractable classical method for approximating high-dimensional classical and quantum systems with unfavorable scaling, and (3) our scalable implementation of 'iterative Bayesian unfolding', an expectation-maximization algorithm for quantum measurement error mitigation, the problem of post-processing results from a quantum computer to account for measurement errors.

Dedication

To my Amma, Appa, and Sneha

Acknowledgements

This dissertation would not have been possible without the incredible research and moral support I have had throughout my Ph.D.

Firstly, I would like to thank my advisor Byron Boots for all his guidance and support during the Ph.D. program. His thoughtful suggestions for research directions, willingness to entertain and debate my frequently half-baked ideas, and generally supportive attitude have been invaluable, especially in the early years. I am also deeply appreciative of his tolerance for my exploration of other fields and ideas.

I would also like to thank Geoff Gordon, Jamie Morgenstern, Sewoong Oh, and Lillian Ratliff for serving on my thesis committee. I'd like to thank Geoff in particular for the sparking some of the ideas that led to this dissertation as well as his collaboration and mentorship during my time at MSR Montreal. I'm also thankful to Jamie for her willingness to spend time advising and mentoring me on totally unrelated research.

I am grateful to my collaborators Sandesh Adhikary, Bibek Pokharel, Jacob Miller, and Guillaume Rabusseau, without whom this work would not have been possible. Sandesh and I collaborated on much of the research that appears in this dissertation, and I'm grateful for his consistent willingness to talk through anything bouncing around my head. In general, I'm fortunate to have had lab mates at both Georgia Tech and the University of Washington who have cultivated an intellectually nourishing environment. I would also like to thank Mark Riedl for his mentorship during my time at Georgia Tech.

I would like to thank my family and friends for being an incredible source of strength throughout my Ph.D. My parents and sister have been supportive and patient with me

even as I've disappeared for extended periods of time immersed in my work. I am grateful for my grandparents Lakshmi, Narasimhan, and Prema, as well as my uncles, aunts, and cousins Varsha, Varun, and Anu for their consistent support and encouragement to pursue a Ph.D. from a young age. I would also like to thank my friends Jake, Maia, Fabiha, Shailee, Emma, Aleina, Akhil, Carla, and many others for their constant words of encouragement. Kara in particular has been a source of tremendous support and an ideal rubber ducky. A Ph.D. program naturally has a lot of ups and downs, but my family and friends have been there to support me through it.

Lastly, I want to acknowledge my late friend and brother Raunak Pednekar. He was always available when I needed someone to talk to, and had a profound impact on my intellectual development, much more than he will ever know. I am grateful to have had him in my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Motivated by shared foundations in linear algebra and probability, the cross-fertilization of ideas between machine learning and quantum information has garnered interest in recent years. This research area is sometimes referred to as 'quantum machine learning' [Schuld et al., 2015, Biamonte et al., 2017], an umbrella term encompassing four distinct directions of research [Schuld and Petruccione, 2021] characterized by their *goals* (whether it tackles classical or quantum problems) and *methods* (whether it uses quantum-inspired classical computation or quantum computation). We can classify machine learning problems as classical or quantum depending on whether the underlying the *data*-generating scheme is classical or quantum, and classify methods as (quantum-inspired) classical or quantum depending on whether the machine learning *model* or *optimizer* requires access to a quantum computer. This work is solely focused on quantum-inspired machine learning methods that can be run on a classical computer (and so we are not focused on identifying scenarios with 'quantum speedups'). Further, we consider applying such methods to both classical and quantum problems. The typical challenges with quantum problems are exponential scaling in model parameters and flat optimization landscape for large systems. This calls for approximation methods that make certain assumptions to make the problem tractable, such as tensor networks, which will feature prominently in our work. Classical problems are a less obvious fit for quantum-inspired methods, although we will explore the particu-

lar case of modeling sequential data with a quantum-inspired extension of hidden Markov models (HMMs), and show that it gives us a model with greater expressiveness and has a practical learning algorithm.

1.1 Main Contributions

The primary research contributions as outlined in this document are:

1. **Hidden Quantum Markov Models**: We develop the theory of hidden quantum Markov models (HQMMs), a quantum-inspired analogue of hidden Markov models (HMMs) for modeling discrete-observation linear stochastic processes. We show that it is the most expressive known model for linear stochastic processes that does not suffer from an undecidable 'negative probability problem'. Along the way, we establish model expressiveness relations with respect to other models of linear stochastic processes proposed in the literature. Using a formulation of HQMMs whose parameterization must satisfy a Stiefel manifold constraint, we propose an iterative maximum likelihood-based learning algorithm to learn HQMMs from data. We test our learning algorithm in several experiments, and find that while HQMMs are theoretically more expressive than HMMs, they have limited advantages over HMMs on our classical datasets. We also formulate kernel HQMMs, where probability distributions are embedded as quantum systems in an RKHS and manipulated using HQMM update rules.

2. **Connections to Tensor Networks**: We investigate the connections between various models for linear stochastic processes and quantum tensor network models, and show that these models are equivalent to the tensor network models in the 'non-terminating limit'. We also study the prospect of parameterizing 'trace-preserving' locally purified density operators, i.e., a tensor network representation of quantum channels that preserves probability and conclude that there are likely insurmountable barriers to achieving such a parameterization.

3. **Measurement Error Mitigation**: Finally, we explore the problem of measurement error mitigation (MEM), a classical post-processing technique for correcting systematic errors inherent to quantum computing hardware. We show that an expectation-maximization algorithm called 'iterative Bayesian unfolding' can be used to scalably implement MEM without returning 'negative probabilities', as is common for other methods in the literature that rely on inverting a stochastic response matrix. We demonstrate our method's potential in mitigating measurement errors on a dataset collected from a 26-qubit Bernstein-Vazirani experiment and a dataset collected from preparing GHZ states up to 127 qubits, both on IBM quantum computers.

1.2 Outline

This document is structured as follows:

- In **Chapter 2**, we present our work on a specific quantum-inspired classical model for classical problems: hidden quantum Markov models (HQMMs). We present the motivation for these models for learning sequential data, describe our previous work in formulating HQMMs and learning them from data, and present an extension to continuous observations called kernel HQMMs.
  - In **Section 2.1**, we present background on hidden Markov models (HMMs).
  - In **Section 2.2**, we give a general discussion of linear stochastic processes and define the notion of model expressiveness we use.
  - In **Section 2.3**, we give some background on predictive state reprsentations (PSRs), discuss the undecidable 'negative probability problem' prohibiting ascertaining whether a candidate PSR will provide negative probabilities as well as their greater expressiveness relative to HMMs, and discuss the existing approaches to learning PSRs and HMMs.
  - In **Section 2.4**, we present some background on norm-observable operator models (NOOMs), a model class that was proposed to avoid the undecidable

negative probability problem by design. We also give some novel results on the expressiveness of NOOM relative to HMMs.

  - In **Section 2.5**, we present our formulation of hidden quantum Markov models (HQMMs) and discuss their expressiveness relative to HMMs, PSRs, and NOOMs, showing that they are the most general known subset of PSRs that do not suffer from the negative probability problem. We also give an iterative maximum-likelihood based learning algorithm to learn the parameters of an HQMM, and demonstrate this algorithm's performance on several datasets.
  - In **Section 2.6**, we develop kernel HQMMs, which embed probability distributions as quantum systems in an RKHS and manipulate these embeddings with HQMM update rules.
  - In **Section 2.7**, we summarize the results presented in this chapter.

- In **Section 3**, we present our work connecting hidden quantum Markov models to tensor networks, a well-studied classical method that attempts to tractably represent high dimensional data applicable to both classical and quantum problems.

  - In **Section 3.1**, we give some preliminaries on quantum tensor networks.
  - In **Section 3.2**, we discuss how HMMS, PSRs, NOOMs, and HQMMs relate to various tensor network models studied in the literature, and show an equivalence to these tensor network models in the 'non-terminating limit'.
  - In **Section 3.3**, motivated by a recent attempt to scale quantum process tomography with tensor networks, we consider the problem of characterizing the 'trace-preserving' constraints that must be imposed on locally purified density operators (LPDOs) so they represent physical quantum channels. We find that there may be insurmountable challenges to finding such a parameterization.

- In **Section 4**, we study a classical approach to mitigating a common problem in quantum computing: measurement errors. We adopt an expectation-maximization

based approach called 'iterative Bayesian unfolding' (IBU) to correct measurement errors and show its potential for scalable measurement error mitigation.

  - In **Section 4.1**, we give some background on the problem of measurement error mitigation in quantum computing and on the 'iterative Bayesian unfolding' approach.
  - In **Section 4.2**, we discuss computational tools and tricks we can use to achieve a scalable implementation of IBU.
  - In **Section 4.3**, we give a derivation showing that IBU is an instantiation of the expectation-maximization algorithm. We also give a Bayesian extension of IBU (which is not itself a real Bayesian method) allowing us to compute the MAP estimate of the error-mitigated distribution.
  - In **Section 4.4**, we demonstrate our scalable IBU implementation on two datasets: an experiment running the 26-qubit Bernstein-Vazirani algorithm and an experiment preparing GHZ states up to 127 qubits.
  - In **Section 4.5**, we summarize our contributions in this chapter.

- In **Section 5**, we make our concluding remarks.

Chapter 2

Hidden Quantum Markov Models

At first glance, the idea of a quantum-inspired classical approach for classical problems might seem gratuitous; how could ideas from quantum mechanics help solve classical problems? In this chapter, we present our motivating scenario of probabilistically modeling discrete sequential data with an extension of hidden Markov models (HMMs) called predictive state representations (PSRs). As we will see, trying to overcome the shortcomings of these models naturally culminates in the formulation and development of quantum-inspired 'hidden quantum Markov models' (HQMMs). This section draws from our work and exposition in Srinivasan et al. [2018b], Srinivasan et al. [2018a], Adhikary et al. [2020], Adhikary et al. [2019], and Srinivasan et al. [2020].

2.1 Hidden Markov Models (HMMs)

Consider the problem of modeling sequential data: given a sequence of observations $y_1, \ldots, y_t$, we wish to predict y_{t+1}. Hidden Markov models [Rabiner, 1986] are a common approach to modeling sequences where an unobserved hidden state undergoes Markovian evolution (where future is independent of past given present) and emits observations at each time-step. In the discrete case (which is what we will consider in this chapter), HMMs maintain and update a *belief state* $\vec{x}$, which is a probability distribution over 'system states' and thus represented by a real, non-negative vector whose entries sum to 1.

Belief state updates are determined by *transition* and *emission* matrices $\mathbf{A}$ and $\mathbf{C}$ of the HMM: each column of the transition matrix describes the transition probabilities to various system states when starting from the system state corresponding to that column, while each column of the emission matrix gives the probabilities over possible observations when the system is in the state corresponding to that column. Since both the transition matrix $\mathbf{A}$ and emission matrix $\mathbf{C}$ are conditional probability tables, they are *column-stochastic* (i.e., the columns sum to 1). Formally, we can define Hidden Markov Models as follows:

Definition 1 (Hidden Markov Model)**.** *An $n-$dimensional Hidden Markov Model for a set of discrete observations $\mathcal{O}$ is a stochastic process given by the tuple $(\mathbb{R}^n, \mathbf{A}, \mathbf{C}, \vec{x}_0)$ where the initial state $\vec{x}_0 \in \mathbb{R}^n$, the transition matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ and the emission matrix $\mathbf{C} \in \mathbb{R}^{|\mathcal{O}| \times n}$ satisfy the following conditions:*

1. ***Non-negative parameters:*** *$\vec{x} \in R^n_{\geq 0}, A \in \mathbb{R}^{n \times n}_{\geq 0}$, and $\mathbb{C} \in \mathbb{R}^{|\mathcal{O}| \times n}$,*
2. ***Normalized initial state:*** *$\|\vec{x}\|_1 = 1$, and*
3. ***Column stochastic operators:*** *$1^T \mathbf{A} = 1^T \mathbf{C} = 1^T$*

These constraints ensure a fully-probabilistic interpretation of the hidden Markov model where the belief state is a probability distribution over hidden system states and the transition and emission matrices are conditional probability tables describing how the system evolves.

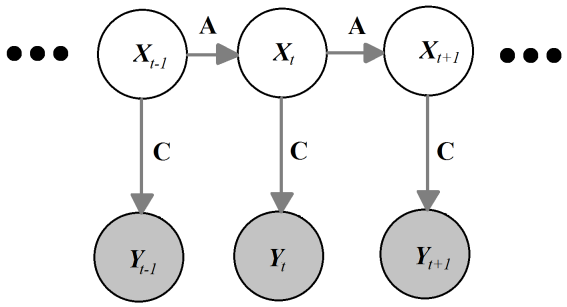


Figure 2.1: A graphical representation of Hidden Markov Models

HMM Equations The state transition update and distribution over observations $\vec{y}$ are given as follows:

$$\vec{x}'_t = \mathbf{A}\vec{x}_{t-1} \qquad \vec{y} = \mathbf{C}\vec{x}'_t \tag{2.1}$$

We can use Bayes' rule to write the belief state vector after conditioning on a given observation y as:

$$\vec{x}_t = \frac{\text{diag}(\mathbf{C}_{(y,:)})\mathbf{A}\vec{x}_{t-1}}{\mathbb{1}^T\text{diag}(\mathbf{C}_{(y,:)})\mathbf{A}\vec{x}_{t-1}} \tag{2.2}$$

where $\text{diag}(\mathbf{C}_{(y,:)})$ is a diagonal matrix with the entries of the yth row of $\mathbf{C}$ along the diagonal, and the denominator is the probability $P(y)$ of observing y and renormalizes the numerator to give a valid probability distribution. We can also compute the probability of a sequence of observations $\bar{y} = y_1, \ldots, y_t$ from a given belief state $\vec{x}$ as follows:

$$P(\bar{y}) = \mathbb{1}^T\text{diag}(\mathbf{C}_{(y_t,:)})\mathbf{A}\cdots\text{diag}(\mathbf{C}_{(y_1,:)})\mathbf{A}\vec{x} \tag{2.3}$$

An alternative representation of HMMs Instead of using the transition and emission matrices $\mathbf{A}$ and $\mathbf{C}$, we can also write the HMM equations above using 'observable' operators (Jaeger [2000]), defines $\mathbf{T}_y = \text{diag}(\mathbf{C}_{(y,:)})\mathbf{A}$. This combines transition and Bayesian updating into a single operator, such that there is a different $\mathbf{T}_y$ for each possible observation y. We can then rewrite Equation 2.2 as:

$$\vec{x}_t = \frac{\mathbf{T}_y\vec{x}_{t-1}}{\mathbb{1}^T\mathbf{T}_y\vec{x}_{t-1}} \tag{2.4}$$

Naturally, the probability of a sequence of observations $\bar{y} = y_1, \ldots, y_t$ starting from belief state $\vec{x}$ is simply:

$$P(\bar{y}) = \mathbb{1}^T\mathbf{T}_{y_t}\ldots\mathbf{T}_{y_1}\vec{x} \tag{2.5}$$

Lastly, note that in this representation, the 1st condition from Definition 1 becomes a requirement that each $\mathbf{T}_y$ be non-negative, and the 3rd condition in Definition 1 becomes $\mathbb{1}^T\sum_y\mathbf{T}_y = \mathbb{1}^T$, i.e., the *sum* of the observable operators (which is also the transition matrix $\mathbf{A}$) must be column stochastic.

As it turns out, the non-negativity requirement on HMM model parameters (condition 1 from Definition 1) limits the expressiveness of HMMs, in the sense that there are finite-dimensional linear stochastic processes that HMMs cannot represent. The standard example is that finite HMMs struggle to capture phenomena where the probability of repeatedly observing the same outcome oscillates (the so-called 'probability clock' [Jaeger, 2000]). Various works in the literature have attempted to overcome this limitation by relaxing the non-negativity condition 1 from the HMM definition and allowing negative valued parameters. Yet, most of these approaches suffered from being either unable to guarantee valid probabilities, or end up greatly restricting model expressiveness in order to ensure valid probabilities. In the next section, we take a closer look at the set of 'linear stochastic processes' and what we mean by model expressiveness in order to discuss the limitations of HMMs and their generalizations.

2.2 Linear Stochastic Processes and Model Expressiveness

Before we can discuss any limitations in HMM expressiveness, we must be precise about what we mean about model expressiveness within the class of (discrete) linear stochastic processes. In service of this, we will need the following definitions:

Definition 2 (Hankel Matrix of a Stochastic Process)**.** *A Hankel matrix* $\mathbf{H}$ *is a bi-infinite matrix where each row corresponds to a past possible sequence of observations, and each row column to a future possible sequence of observations.*

$$\mathbf{H} = \begin{pmatrix} P(y_f^{(1)}|y_p^{(1)}) & P(y_f^{(2)}|y_p^{(1)}) & \dots & P(y_f^{(j)}|y_p^{(1)}) & \dots \\ P(y_f^{(1)}|y_p^{(2)}) & P(y_f^{(2)}|y_p^{(2)}) & \dots & P(y_f^{(j)}|y_p^{(2)}) & \dots \\ \vdots & \vdots & \ddots & \vdots & \ddots \\ P(y_f^{(1)}|y_p^{(i)}) & P(y_f^{(2)}|y_p^{(i)}) & \dots & P(y_f^{(j)}|y_p^{(i)}) & \dots \\ \vdots & \vdots & \ddots & \vdots & \ddots \end{pmatrix} \tag{2.6}$$

As visualized in Equation 2.6, the Hankel matrix is a complete description of the stochastic process: it specifies the probabilities for every combination of past and future

observations. Note that the rows and columns index past and future observations of *any* length, so there is no row- or column-wise normalization constrint (also the matrix is bi-infinite). This is a well-studied construct in the literature [Jaeger, 2000, Balle et al., 2014a, Thon and Jaeger, 2015], sometimes referred to as the system-dynamics matrix [Singh et al., 2004]. These works are further primarily interested in *linear* stochastic processes, which can be defined as follows:

Definition 3 (Linear Stochastic Process)**.** *A stochastic process is linear if its Hankel matrix has finite rank.*

Modeling the set of such processes will be our primary interest, and we will see shortly, HMMs represent a subset of linear stochastic processes. To build some intuition on these processes, we draw from the exposition in [Singh et al., 2004]. First, we note that the full Hankel matrix can be reconstructed from linear combinations of a finite number of columns of $\mathbf{H}$. For a Hankel matrix of rank k, Singh et al. [2004] arrive at a notion of 'predictive state representation' of linear stochastic processes by reasoning as follows: let $\mathbf{X}$ be an $\infty \times k$ submatrix of $\mathbf{H}$ consisting of all the rows and a linearly independent set of columns of $\mathbf{H}$. Then, by the finite rank of $\mathbf{H}$, we can write any column of $\mathbf{H}$ (i.e., conditional probability of future observation sequence $y_f^{(j)}$ given *all possible* past observation sequences $\vec{y}_p$) as $P(y_f^{(j)}|\vec{y}_p) = \mathbf{X}\vec{t}_j$ for some vector of weights $\vec{t}_j$. Additionally, any element of $\mathbf{H}$ (i.e., conditional probability of any future observations given *a particular* past observation sequence) is simply $P(y_f^{(j)}|y_p^{(i)}) = \mathbf{X}_{(i,:)}\vec{t}_j$. Note that the row $\mathbf{X}_{(i,:)}$ is a *sufficient statistic* of history $y_p^{(i)}$ as it summarizes all necessary information about the past to predict the future. Singh et al. [2004] refer to this summary statistic as a 'predictive state' and show how to update it on observation, thereby formulating a 'predictive state representation' (PSR)[1] *model* of linear stochastic processes (we will give a formal definition shortly). For now, we note that linear stochastic processes and their representations have been studied from different angles and go by different names, including quasi-realizations [Vidyasagar, 2011], stochastic weighted automata [Balle et al., 2014a], observable operator models [Thon and

[1]Singh et al. [2004] use PSRs to refer to both controlled and uncontrolled models; we use it in this document to only refer to uncontrolled models.

Jaeger, 2015]. We refer to them in this work as predictive state representations [Singh et al., 2004], and we develop them in greater detail in Section 2.3.

Model Expressiveness Having established that we will be working within the class of linear stochastic processes, we will explore the expressiveness of various *model classes* for modeling these processes. By *model* we mean a set of parameters that can assign probabilities to any future observation sequence given any past sequence of observations, and thus (in principle) generate the Hankel matrix of some linear stochastic process. By *model class* we mean a set of models whose parameters obey the same set of constraints and generate probabilities with the same set of equations. Thus, a model class refers to a family of models, and a model is a particular instantiation of some model class. In this sense, hidden Markov models are a model class (as are predictive state representations (PSRs), which we develop in Section 2.3).

We are particularly interested in whether the constraints imposed by a model class in order to guarantee generation of a Hankel matrix with valid probabilities (such as those on HMMs in Definition 1) necessarily limit its expressiveness, and if so, to what extent. To this end, we give the following definition of 'expressiveness of model classes':

Definition 4 (Expressiveness of Model Classes)**.** *Given model classes $\mathcal{A}, \mathcal{B}$ for modeling linear stochastic processes, we say $\mathcal{A}$ is more expressive than $\mathcal{B}$ if the set of Hankel matrices representable by all finite-dimensional models in $\mathcal{B}$ is a strict subset of the set of Hankel matrices representable by all finite-dimensional models in $\mathcal{A}$. In such a case, we write $\mathcal{B} \subset \mathcal{A}$.*

In other words, when $\mathcal{B} \subset \mathcal{A}$, every model in $\mathcal{B}$ has an equivalent parameterization in $\mathcal{A}$ (in the sense of generating the same Hankel matrix) but the opposite is not true. It is worth noting that this notion of expressiveness says nothing about comparing models with the same number of dimensions or *compactness* of representation; it just requires a model in $\mathcal{B}$ to be representable by a finite dimensional model in $\mathcal{A}$, no matter how many dimensions. Indeed, we will see a case where a compact model class has limited

expressiveness. Additionally, non-linear models may be able to represent the same process more compactly than linear models. While we remark on (linear) model class' compactness where relevant, it is a separate question and not central to our discussion.

2.3 Predictive State Representations (PSRs)

We laid the groundwork in the previous section to show that predictive state representations are a model class that can naturally be derived from finite rank Hankel matrices of linear stochastic processes. We complete the construction of PSRs and also provide a general definition that abstracts away the details of this derivation and shows their close similarity to HMMs. Since PSRs are capable of representing any finite-rank Hankel matrices, this essentially demonstrates that PSRs are the most general model class for representing linear stochastic processes. A deeper discussion of how HMMs compare to PSRs in terms of expressiveness is discussed in Section 2.3.3.

As discussed in the previous section, the submatrix $\mathbf{X}_{\infty\times k}$ of the rank-k Hankel matrix $\mathbf{H}_{\infty\times\infty}$ consists of k linearly independent columns and can reconstruct the probability of any future observation sequence $y_f^{(j)}$ given any past observation sequence $y_p^{(i)}$. The k future observation sequences corresponding to the selected columns of $\mathbf{H}$ are called 'core tests' [Singh et al., 2004] or 'characteristic events' [Jaeger, 2000], and we denote the vector of k core test sequences as $\vec{y}^{(k)}$. Further, a row $\vec{x}_i \in \mathbb{R}^{1\times k}$ of the submatrix $\mathbf{X}$ corresonds to a particular sequence of past observations $y_p^{(i)}$, and there exist some weights $\vec{t}_j \in \mathbb{R}^{k\times 1}$ that give the probability of any conditional future observation sequence $P(y_f^{(j)}|y_p^{(i)}) = \vec{x}_i \cdot \vec{t}_j$. The row $\vec{x}_i$ is then a 'predictive state representation' of past observation sequence $y_p^{(i)}$, as it captures all the information about the past needed to predict the future. With this notion of state, previous works [Singh et al., 2004, Thon and Jaeger, 2015] also develop a notion of 'conditioning on observation' which we reproduce here. Given some observation o, we want to update the predictive state $\vec{x}_i$ summarizing history $y_p^{(i)}$ to a predictive state $\vec{x}_{(o,i)}$ summarizing past observations $y_p^{(o,i)}$. First, we collect the weight vectors for k future observation sequences consisting of observation o followed by each of the k core tests into

a matrix $\boldsymbol{\tau}_o = [\vec{t}_{(j_1,o)}, \ldots, \vec{t}_{(j_k,o)}]_{k\times k}$. Observe that this allows us to write the probability of each of the core tests following observation o as $P(\vec{y}_f^{(k,o)}|y_p^{(i)}) = \vec{x}_i^T \boldsymbol{\tau}_o$. We can also write the probability of observation o given the past $y_p^{(i)}$ as $P(y_f^{(o)}|y_p^{(i)}) = \vec{x}_i \cdot \vec{t}_o$, but conceptually, this vector is simply a marginal of the matrix $\boldsymbol{\tau}_o$ (marginalizing out the core tests). Thus, we can use a 'generalized marginalization vector' $\vec{\sigma} \in \mathbb{R}^{k\times 1}$ (analogous to the ones vector $\mathbb{1}$ for marginalization in HMMs in Equation 2.5 to write $\vec{t}_o = \boldsymbol{\tau}_o \vec{\sigma}$. Finally, we can use Bayes' rule to determine the update rule for predictive state representations after observation o (which looks nearly identical to that of HMMs with observable operators in Equation 2.4):

$$P(y_f^{(j)}|y_p^{(o,i)}) = \frac{P(y_f^{(j,o)}|y_p^{(i)})}{P(y_f^{(o)}|y_p^{(i)})} = \frac{\vec{x}_i^T \boldsymbol{\tau}_o}{\vec{x}_i^T \boldsymbol{\tau}_o \vec{\sigma}} \tag{2.7}$$

We use the above derivation of updating predictive states to give the following formal definition of PSRs [Srinivasan et al., 2020, Thon and Jaeger, 2015] (where we have transposed the vectors and matrices from the derivation above):

Definition 5 (Predictive State Representations)**.** *An $n-$dimensional predictive state representation for a set of discrete observations $\mathcal{O}$ is a stochastic process given by the tuple $(\mathbb{R}^n, \vec{\sigma}, \{\tau_y\}_{y\in\mathcal{O}}\}, \vec{x}_0)$ where the initial state $\vec{x}_0 \in \mathbb{R}^n$ (consisting of sufficient statistics of history), observable operators $\tau_y \in \mathbb{R}^{n\times n}$, and linear evaluation functional $\vec{\sigma} \in \mathbb{R}^n$ satisfy the following conditions:*

1. ***Normalized initial state**: $\vec{\sigma}^T \vec{x}_0 = 1$,*
2. ***Normalized marginal over observations**: $\vec{\sigma}^T \sum_y \tau_y = \vec{\sigma}^T$, and*
3. ***Valid probabilities generated**: $P(y_T, \ldots, y_1) = \vec{\sigma}^T \tau_{y_T} \ldots \tau_{y_1} \vec{x} \geq 0$ for any arbitrary-length sequence $y_1, \ldots, y_T \in \mathcal{O}^{\otimes T}$.*

This definition is nearly identical to that of HMMs; the primary difference to note is that the non-negativity requirement on HMM parameters (condition 1 of Definition 1) has been replaced with a non-constructive requirement on PSR parameters that they simply

produce non-negative probabilities (condition 3 in Definition 5); so PSRs are essentially HMMs whose parameters may be negative-valued. The state update equation and probability computation for PSRs are also nearly identical to that of HMMs in the observable operator representation (Equations 2.4, 2.5):

$$\vec{x}_t = \frac{\tau_y \vec{x}_{t-1}}{\vec{\sigma}^T \tau_y \vec{x}_{t-1}} \qquad P(\bar{y}) = \vec{\sigma}^T \tau_{y_t} \dots \tau_{y_1} \vec{x} \tag{2.8}$$

There is also the question of whether a given set of PSR parameters represents a unique linear stochastic process. Previous works [Ito et al., 1992, Vidyasagar, 2011, Thon and Jaeger, 2015] show that every minimal[2] PSR is only unique up to a similarity transform:

Theorem 1. *Two minimal PSR representations* $(\mathbb{R}^n, \vec{\sigma}, \{\tau_y\}_{y\in\mathcal{O}}\}, \vec{x}_0)$ *and* $(\mathbb{R}^n, \vec{\sigma}', \{\tau'_y\}_{y\in\mathcal{O}}\}, \vec{x}'_0)$ *are equivalent (i.e., they represent the same linear stochastic process) if and only if there exists some non-singular* $\mathbf{S} \in \mathbb{R}^{n\times n}$ *such that* $\mathbf{S}^{-1}x_0 = x'_0$, $\mathbf{S}\tau_y\mathbf{S}^{-1} = \tau'_y$, *and* $\vec{\sigma}^T\mathbf{S} = \vec{\sigma}'^T$.

This can be seen from the fact that the probabilities assigned to any sequence (as in Equation 2.8) by a PSR is the same as under the similarity-transformed PSR. This equivalence will play an important role in establishing the expressiveness relationships between various model classes.

2.3.1 The Negative Probability Problem

Observe that condition 3 in Definition 5 of PSRs simply asserts that valid PSR parameters 'do the right thing' in terms of linearly updating predictive states and generating valid (non-negative) probabilities *without describing how to construct such models.* This is in contrast to HMMs, which guarantee valid, non-negative probabilities by explicitly requiring the observable operators to be non-negative.

Indeed, it is not immediately obvious how to construct a valid PSR or verify whether a given candidate PSR satisfies condition 3 in Definition 5 since the definition is non-constructive. It turns out that for given candidate PSR $(\mathbb{R}^n, \vec{\sigma}, (\tau_y)_{y\in\mathcal{O}}, \vec{x}_0)$ satisfying

[2]Minimal here means that the PSR is in its most compact representation; it uses only as many 'core tests' as the rank of the Hankel matrix it represents and no more.

conditions 1 and 2, the question of whether it will violate condition 3 is *undecidable* [Denis and Esposito, 2004, Wiewiora, 2008]; this the cost of discarding the positivity constraint from HMMs to allow negative-valued PSR parameters, and the root of the infamous negative probability problem (NPP) in PSRs. This poses a particular challenge for learning PSRs from data: how can we use the estimated model if it might assign negative probabilities to predictions and we cannot determine a priori whether this will happen? In the special case where the estimated parameters happen to be positive, we have a sufficient condition for generating valid probabilities, namely that the PSR is a hidden Markov model. But in the general case, we simply cannot guarantee that we have learnt *valid* PSR model parameters satisfying condition 3 of the definition. Indeed, Hsu et al. [2012] and Siddiqi et al. [2010] present a statistically consistent spectral algorithm to learn the parameters of a PSR by writing moments of the distribution over observations as a function the model parameters, so empirically estimating the moments and inverting the equations gives the model parameters (up to a similarity transform). Although the algorithm is statistically consistent, in practice the model parameters are estimated with finite data and the estimated model does predict negative probabilities. Thus, the undecidable negative probability problem is a hindrance to learning valid PSRs.

There are mainly two practical approaches to handling the negative probability problem: either we manage the problem with heuristics (e.g. by only filtering on short sequences for which the probabilities are non-negative, projecting the learned parameters to the nearest HMM, or by reversing the sign or zeroing negative probabilities and renormalizing, [Balle et al., 2014b, Cohen et al., 2013]), or we use a model class that avoids the NPP by construction. HMMs are the canonical example of the latter approach, but as we will see their expressiveness is limited. The search for a more expressive model class that avoids the NPP led to the development of 'norm-observable operator models' (NOOMs) (which we show are also limited in the expressiveness), and culminates in our development and formulation of HQMMs.

2.3.2 A Convex Cone Characterization of PSRs

Despite the NPP and non-constructive definition of PSRs, previous work [Heller, 1965, Jaeger, 2000] has been able to provide a geometric interpretation of condition 3 in Definition 5, characterizing the valid states for arbitrary PSRs as a convex cone. Although such an interpretation does not help overcome the NPP by virtue of its undecidability, it will help with comparisons of model expressiveness. We state this geometric interpretation below:

Theorem 2. *A candidate PSR* $(\mathbb{R}^n, \vec{\sigma}, \{\tau_y\}_{y\in\mathcal{O}}\}, \vec{x}_0)$ *never generates negative probabilities if and only if there is a pointed convex cone* K *such that:*

1. ***Initial state is in the cone:*** $\vec{x}_0 \in K$
2. ***Cone is closed under operators*** τ_y*:* $\tau_y K \subseteq K$ *for all* y
3. ***All points in the cone generate valid probabilities:*** $\vec{\sigma}^T \vec{x} \geq 0$ *for all* $\vec{x} \in K$

Thus, a useful conceptual characterization of a candidate PSR with parameters $\{\tau_y\}_{y\in\mathcal{O}}$ is the convex cone of valid initial states it admits, i.e., the initial states that can be safely updated without ever assigning a negative probability to a future observation. Conditions 1 and 2 in Theorem 2 guarantee that any initial state inside such a cone will stay inside the cone under action by τ_y, and condition 3 guarantees that any state inside the cone will evaluate to a non-negative probability. If there is no such cone, the model is not a valid PSR.

2.3.3 PSRs vs. HMMs

As we have suggested throughout this chapter, PSRs are a more expressive model class than HMMs, (i.e., HMM $\subset$ PSR). On the other hand, the NPP means that identifying whether a set of PSR parameters is valid is undecidable, while the non-negativity of HMMs enables easy verification of their validity. We explore the expressiveness relationship in more depth here.

View from Hankel Matrices The natural derivation of PSRs from the Hankel matrix of linear stochastic processes showed that any rank-k linear stochastic process can always be represented by a rank-k PSR. Thus, any linear stochastic process can be represented by some (finite) PSR. Additionally, the Hankel matrix rank also determines the minimal state dimension needed to represent the PSR. Extensive previous work [Heller, 1965, Ito et al., 1992, Jaeger, 2000, Singh et al., 2004] has identified that PSRs are the most expressive and compact (linear) model of linear stochastic processes.

The key insight here is this: the operators τ_y of a PSR come from the weights $\vec{t}_j$ that specify the linear combination of the columns of the submatrix $\mathbf{X}$ (which correspond to core tests) to recover column j of the Hankel matrix $\mathbf{H}$. For PSRs, these weights can be non-negative, so linear stochastic processes requiring *any* linear combinations of the columns of $\mathbf{X}$ can be represented. HMMs on the other hand only allow non-negative valued operators, and so only linear stochastic processes where columns j of $\mathbf{H}$ can be reconstructed with non-negative weights $\vec{t}_j$ can be represented by HMMs. This helps explain why HMM $\subset$ PSR.

The Positive Realization Problem The general question of which PSRs have equivalent finite-dimensional HMMs is known as the *positive realization problem* [Benvenuti and Farina, 2004, Vidyasagar, 2011]. Heller [1965] and Jaeger [2000] have an answer via the convex cone characterization of PSRs:a PSR has an equivalent HMM representation (via similarity transform, as established by Theorem 1) if and only if the convex cone characterizing the PSR is k-polyhedral for some finite k, i.e., it is generated by a finite set of k vectors. The fact that PSRs admit infinitely generated convex cones while HMMs do not is another way to view their superior expressiveness relative to HMMs.

Probability Clock While the undecidable NPP prohibits verifying the validity of a given PSR, Theorem 2 does give us a recipe to *find* valid PSRs that do not suffer from the NPP: select a desired convex cone and then construct operators such that the cone is closed under their application. Jaeger [2000] used this recipe and the fact that HMMs do not

(a) A polyhedral cone (b) An infinitely generated cone

Figure 2.2: Visualization of polyhedral cones (left) of valid initial states $\vec{x}_0$ which both PSRs and HMMs admit, and infinitely generated cones (right) which only PSRs admit

admit infinitely generated cones to pick an infinitely generated cone of valid initial states, and construct operators under which the cone is closed. They refer to the constructed PSR as the 'probability clock' (since the probability of repeatedly observing the same observation oscillates) and it serves as the canonical example of a finite dimensional PSR which cannot be represented by any finite-dimensional HMM.

The Perron-Frobenius Theorem The greater expressiveness of PSRs can also be understood through the Perron-Frobenius theorem [Adhikary et al., 2019], which states that the largest eigenvalue of a real-valued matrix must be real. For HMMs, this prohibits the system state from oscillating in response to repeatedly observing the same observation y and applying operator $\mathbf{T}_y$; such oscillatory behaviour requires the largest eigenvalue of $\mathbf{T}_y$ to be complex. On the other hand, negative-valued PSR operators τ_y can have complex largest eigenvalues, and this explanation suggests that the superior expressiveness of PSRs over HMMs comes from their ability to capture such oscillatory behaviours.

2.3.4 Learning PSRs and HMMs

The well-known Baum-Welch algorithm [Rabiner, 1986, Bilmes et al., 1998] is the standard approach for learning HMMs: it is an application of the iterative expectation-maximization algorithm that seeks to find parameters that (locally) maximize the log likelihood. However, the algorithm is relatively slow and prone to getting stuck in local optima; this along with the lesser expressiveness of HMMs prompted work into learning algorithms for PSRs.

Jaeger [2000] and Jaeger et al. [2005] were early proposals for learning PSRs (as 'OOMs'), but they required manual selection of core tests. Hsu et al. [2012] developed a statistically consistent spectral learning algorithm for learning HMMs of a given dimension. Siddiqi et al. [2010] extended this algorithm to 'reduced-rank HMMs' which could learn HMMs in their minimal PSR representation (up to a similarity transform). Later, Anandkumar et al. [2012] showed how the transition and emission operators of an HMM can be recovered through a tensor decomposition approach. Most recently, Hefny et al. [2015] propose a 2-stage least squares method which first learns predictive state representations and then learn a mapping between predictive states. These methods use singular value decompositions to identify the linearly independent subspace of core tests (under the assumption of 1-step observability), hence manual selection is not needed. The algorithms also take a method of moments approach, writing moments of the distribution over observations as a function the model parameters, so empirically estimating the moments and inverting the equations gives PSR model parameters (which may be a similarity transform away from valid HMM parameters). However, the matrix inversion makes the model prone to assigning negative probabilities, which is exacerbated when we have limited data. There has been some empirical investigation in handling negative probabilities with heuristics [Cohen et al., 2013], using the output of the spectral algorithm as an initialization for HMMs [Balle et al., 2014b] or optimizing such outputs via exterior point methods [Shaban et al., 2015]. However, heuristic projections break the theoretical guarantees of convergence while exterior point methods nudge the solution towards explicitly non-negative parameters which may not be close to the optimal similarity-transformed HMM solution and also nullify any benefits of higher expressiveness of PSRs. These empirical investigations of EM and the spectral algorithm for learning HMMs/PSRs and generally find that the benefits of initializing with the spectral solutions are unclear and that EM produces better solutions than the spectral algorithms (albeit at higher computational cost).

2.4 Norm-observable Operator Models (NOOMs)

The only alternative to using heuristics, projections, or exterior point methods for handling the negative probability problem is to use a model class that altogether avoids the NPP by design. We know that HMMs are an option, but we have seen their limited expressiveness manifest as an inability to capture 'oscillating' probabilities (see Section 2.3.3). This naturally leads to the question: can we design a model class for linear stochastic processes that is more expressive than HMMs, does not suffer from the negative probability problem, and whose parameters can be learned from data?

Motivated by this question, Zhao and Jaeger [2007] and Zhao and Jaeger [2010] introduce norm-observable operator models or NOOMs. The central idea is to wrap the output of the model with the non-linear function $\|\cdot\|^2$ so that it always returns non-negative values.

Definition 6 (NOOMs [Zhao and Jaeger, 2010])**.** *An n-dimensional Norm Observable Operator Model with a set of discrete observations $\mathcal{O}$ is a tuple $(\mathbb{R}^n, (\phi_y)_{y\in\mathcal{O}}, \vec{\psi}_0)$ where initial state $\vec{\psi}_0 \in \mathbb{R}^n$ and observable operators $\{\phi_y\}_{y\in\mathcal{O}} \in \mathbb{R}^{n\times n}$ satisfy the following conditions:*

1. ***Normalized initial state:*** $\|\vec{\psi}_0\|_2 = 1$
2. ***'Norm-preserving' Operators:*** $\sum_{y\in\mathcal{O}} \phi_y^T \phi_y = \mathbb{I}$

These models avoid the NPP by using the 2-norm of the state to recover a valid probability which, unlike HMMs, is insensitive to the use of negative parameters in the matrices $\boldsymbol{\phi}_y$. The updated state after observing $y \in \mathcal{O}$ and the probability of that observation can be computed as:

$$\vec{\psi}_t = \frac{\phi_y \vec{\psi}_{t-1}}{\|\phi_{y_t} \dots \phi_{y_1} \vec{\psi}\|^2} \qquad P(\bar{y}) = \|\phi_{y_t} \dots \phi_{y_1} \vec{\psi}\|^2 \tag{2.9}$$

2.4.1 NOOMs vs {PSRs, HMMs}

Zhao and Jaeger [2010] claim that all stochastic processes can be represented as a NOOM

in some inner product space, but this space may be infinite dimensional. For our purposes, we care about the expressiveness of *finite*-dimensional NOOMs. On this question, Zhao and Jaeger [2007] demonstrated that NOOMs $\not\subseteq$ HMMs using a 'NOOM probability clock', where the latent state (and therefore conditional probabilities) exhibit oscillatory behavior (since negative entries in NOOM operators allow their top eigenvalues to be complex valued). The NOOM probabilitiy clock is a constructive example of a finite-dimensional NOOM that has no equivalent finite-dimensional HMM. Zhao and Jaeger [2010] further show that despite the apparent non-linear expression, every n-dimensional NOOMs has an equivalent n^2 dimensional PSR parameterization, i.e., NOOMs $\subseteq$ PSRs. We reproduce their result NOOM $\subseteq$ PSR here, and this style of argument plays a role in our later results on the expressiveness of HQMMs.

Theorem 3 (*NOOMs $\subseteq$ PSRs*)**.** *Every n-dimensional NOOM has an equivalent representation as an n^2-dimensional PSR.*

Proof. We write the joint probability of a sequence as computed by a NOOM and manipulate it using using the relationship between 2-norm and trace as follows:

$$\begin{aligned} P(y_1, \ldots, y_T) &= \left|\left| \phi_{y_T} \ \cdots \ \phi_{y_1} \vec{\psi}_0 \right|\right|_2^2 \\ &= \mathrm{tr}(\phi_{y_T} \ \cdots \ \phi_{y_1} \vec{\psi}_0 \vec{\psi}_0^T (\phi_{y_1})^T \ \cdots \ (\phi_{y_N})^T) \\ &= \vec{\mathbb{I}}_{n^2}^T (\phi_{y_T} \otimes \phi_{y_T}) \ldots (\phi_{y_1} \otimes \phi_{y_1}) \left(\vec{\psi}_0 \otimes \vec{\psi}_0 \right) \\ &= \vec{\sigma}^T \tau_{y_T} \ \cdots \ \tau_{y_1} \ \vec{\rho}_0 \end{aligned} \tag{2.10}$$

where $\tau_y = \phi_y \otimes \phi_y \in \mathbb{R}^{n^2 \times n^2}$, $\vec{\rho}_0 = \vec{\psi}_0 \otimes \vec{\psi}_0 \in \mathbb{R}^{n^2 \times 1}$, and $\vec{\sigma} = \vec{\mathbb{I}}$ is the vectorized identity matrix. This makes every NOOM a PSR: condition 1 of Definition 5 of PSRs is satisfied since $\vec{\sigma}^T \vec{x}_0 = \mathrm{tr}(\vec{\psi}_0 \vec{\psi}_0^T) = \|\psi_0\|^2 = 1$; condition 2 of the PSR definition is satisfied since $\vec{\mathbb{I}}^T \sum_y (\phi_y \otimes \phi_y) = \mathrm{vec}\left(\sum_y \phi_y^T \mathbb{I} \phi_y \right)^T = \vec{\mathbb{I}}^T$; condition 3 of the PSR definition is satisfied since the probability computation in Equation 2.10 uses the squared 2-norm and hence is non-negative. Thus, every finite-dimensional NOOM can be written as a finite-dimensional PSR $(\mathbb{R}^{n^2}, \ \vec{\mathbb{I}}, \ \{\phi_y \otimes \phi_y\}_{y \in \mathcal{O}}, \vec{\psi}_0 \otimes \vec{\psi}_0)$ and NOOMs are no more expressive than PSRs. $\square$

An interesting feature of NOOMs is that they can admit a more compact representation of than the equivalent PSR parameterization, since they use the non-linear function $\|\cdot\|^2$ to recover the probability of a sequence; for instance, the 2-dimensional 'NOOM probability clock' corresponds to a 3-dimensional PSR.

Although Zhao and Jaeger [2007] established that NOOMs $\subseteq$ PSRs, they left open the question of whether NOOMs are *just as* expressive as PSRs, (i.e., whether NOOMs = PSRs), and how NOOMs compare to HMMs. We answered these questions in Srinivasan et al. [2020], and reproduce our results here. Before doing so, we give the definition of a Schmidt decomposition (similar to a singular value decomposition), which plays an essential role in our results:

Definition 7 (Schmidt Decomposition and Schmidt Rank)**.** *Given Hilbert spaces $\mathcal{H}_1$ and $\mathcal{H}_2$, for any $X \in \mathcal{H}_1 \otimes \mathcal{H}_2$ there exist two sets of orthonormal vectors $\{u_1, \ldots, u_k\} \in \mathcal{H}_1$ and $\{v_1, \ldots, v_k\} \in \mathcal{H}_2$, and real, non-negative scalars λ_i such that $X = \sum_{i=1}^{k} \lambda_i u_i \otimes v_i$. The scalars λ_i (called Schmidt coefficient) are unique up to ordering, and the Schmidt-rank of X is the number of Schmidt coefficients.*

The relevance of this result is that we can now describe the PSRs that have equivalent NOOMs as those where the evaluation functional $\vec{\sigma} = \vec{\mathbb{I}}$, and the operators and state have a rank-1 Schmidt decomposition $\tau_y = \phi_y \otimes \phi_y$, $\vec{x}_0 = \vec{\psi}_0 \otimes \vec{\psi}_0$. If the Schmidt-rank of a given set of PSR parameters (which may be HMM parameters too) is greater than 1, it cannot have an equivalent NOOM.

Our first result below is that HMM $\not\subseteq$ NOOM [Srinivasan et al., 2020]; so although the NOOM probability clock demonstrates that NOOMs can capture processes with the sort of oscillatory behaviours that PSRs struggle with, there are also processes that HMMs can represent in finite dimensions that NOOMs cannot. The proof relies on the fact that both NOOMs and HMMs are special cases of PSRs, and any two equivalent PSRs must be related by a similarity transform (Theorem 1). We show that there is no similarity transform between HMMs and NOOMs that preserves the normalization requirement of NOOM states.

Theorem 4. *[HMMs $\not\subseteq$ NOOMs] There exist finite-dimensional hidden Markov models that have no equivalent finite-dimensional norm-observable operator model.*

Proof. We give a proof by contradiction by constructing a set of HMMs for which there can be no equivalent NOOMs. Let $\mathcal{A} = (\mathbb{R}^p, \vec{\sigma}, \{\tau_y\}_{y\in\mathcal{O}}, \vec{x}_0)$ be a minimal PSR equivalent to some hidden Markov model $\mathcal{M} = (\mathbb{R}^m, \mathbf{A}, \mathbf{C}, \vec{p}_0)$ for which some future state reachable from the initial state can be written as a convex combination of some previously reached states, i.e., $\vec{x}_k = \alpha\vec{x}_i + \beta\vec{x}_j$ for some $\alpha, \beta > 0$, $\alpha+\beta = 1$ and some $i, j, k \in \mathbb{N}$ with $i < j < k$ and $\vec{x}_k = \frac{\tau_{y_k}\cdots\tau_{y_1}\vec{x}_0}{\vec{\sigma}^T\tau_{y_k}\cdots\tau_{y_1}\vec{x}_0}$ for some sequence of observations $Y = y_1, \ldots, y_k$ (and similarly for $\vec{x}_j$ and $\vec{x}_i$ which truncate the observations at y_j and y_i respectively).

Suppose there exists an equivalent NOOM (represented in its vectorized form) $\mathcal{M}' = (\mathbb{R}^n, \vec{\mathbb{I}}, \{\boldsymbol{\phi}_y\}, \vec{\psi}_0)$. Let $\mathcal{A}' = (\mathbb{R}^p, \vec{\sigma}', \{\tau'_y\}_{y\in\mathcal{O}}, \vec{x}'_0)$ be a minimal PSR computing the same distribution as $\mathcal{M}'$.

Then, by Theorem 1, we have some similarity transform $\mathbf{S}$ such that $\vec{\sigma}'^T = \vec{\sigma}^T\mathbf{S}$, $\tau'_y = \mathbf{S}^{-1}\tau_y\mathbf{S}$, and $\vec{x}'_0 = \mathbf{S}^{-1}\vec{x}_0$. Thon and Jaeger [2015] show that there are matrices Φ, Π that relate the NOOM to its minimal representation as $\mathcal{A}' = \Pi\Phi^+\mathcal{M}'\Phi\Pi^+$ (where $+$ represents the Moore-Penrose pseudoinverse). This allows us to relate the NOOM with the minimal PSR representation of its equivalent HMM as $\vec{\mathbb{I}}^T\Phi\Pi^+\mathbf{S}^{-1} = \vec{\sigma}^T$, $\tau_y = \mathbf{S}\Pi\Phi^+\boldsymbol{\phi}_y\Phi\Pi^+\mathbf{S}^{-1}$, and $\vec{x}_0 = \mathbf{S}\Pi\Phi^+\vec{\psi}_0$.

Now, by the NOOM evolution rules, the NOOM state at timestep k for the sequence Y is $\vec{\psi}_k = \frac{\boldsymbol{\phi}_{y_k}\cdots\boldsymbol{\phi}_{y_1}\vec{\psi}_0}{\vec{\mathbb{I}}^T\boldsymbol{\phi}_{y_k}\cdots\boldsymbol{\phi}_{y_1}\vec{\psi}_0}$ and the probability of any given observation at that time-step is $P(y|\vec{\psi}_k) = \vec{\mathbb{I}}^T\boldsymbol{\phi}_y\vec{\psi}_k$. Further, note that the NOOM state must be a vectorized rank-1 matrix whose eigenvector has unit ℓ_2 norm, i.e., $\vec{\psi}_k = \mathrm{vec}(\vec{\psi}_k\vec{\psi}_k^T)$ with $\|\vec{\psi}_k\|_2 = 1$.

But we can also write NOOM state in terms of its equivalent HMM as

$$\begin{aligned}
\vec{\psi}_k &= \Phi\Pi^+\mathbf{S}^{-1}\vec{x}_k \\
&= \Phi\Pi^+\mathbf{S}^{-1}(\alpha\vec{x}_i + \beta\vec{x}_j) \\
&= \Phi\Pi^+\mathbf{S}^{-1}\left(\alpha\left(\mathbf{S}\Pi\Phi^+\vec{\psi}_i\right) + \beta\left(\mathbf{S}\Pi\Phi^+\vec{\psi}_j\right)\right) \\
&= \alpha\vec{\psi}_i + \beta\vec{\psi}_j
\end{aligned} \tag{2.11}$$

If $\vec{\psi}_i$ and $\vec{\psi}_j$ are valid NOOM states, we have that $\vec{\psi}_k = \alpha\vec{\psi}_i + \beta\vec{\psi}_j = \alpha\text{vec}(\vec{\psi}_i\vec{\psi}_i^T) + \beta\text{vec}(\vec{\psi}_j\vec{\psi}_j^T) = \text{vec}(\alpha\vec{\psi}_i\vec{\psi}_i^T + \beta\vec{\psi}_j\vec{\psi}_j^T)$. However, $\alpha\vec{\psi}_i + \beta\vec{\psi}_j$ is not the vectorization of a rank-1 matrix in general. In particular, $\vec{\psi}_k$ has unit Schmidt-rank only if $\vec{\psi}_i$ and $\vec{\psi}_j$ are linearly dependent, which is true only if $\vec{x}_i$ and $\vec{x}_j$ are linearly dependent. Thus, whenever $\vec{x}_i$ and $\vec{x}_j$ are linearly independent, $\vec{\psi}_k$ cannot have unit Schmidt-rank. But as normalized HMM states, $\vec{x}_i$ and $\vec{x}_j$ are always linearly independent, unless they are exactly the same. Hence, a contradiction. Thus, for such an HMM, there is no equivalent NOOM. □

NOOMs are a restrictive model class The proof above essentially argues that if an HMM had an equivalent NOOM, then anytime a reachable HMM state can be written as a convex combination of some other reachable states, the equivalent NOOM state should also admit a representation as a convex combination of the NOOM-equivalent reachable states, but such a representation violates the condition that NOOM states have unit Schmidt-rank, and hence there cannot be an equivalent NOOM. Below, we provide a concrete example of an HMM where a future state is a linear (and convex) combination of two prior linearly independent states:

$$\vec{x}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \tau_1 = \begin{bmatrix} 0.25 & 0.5 \\ 0.75 & 0 \end{bmatrix} \quad \tau_2 = \begin{bmatrix} 0 & 0 \\ 0 & 0.5 \end{bmatrix} \quad \vec{\sigma} = \mathbb{1} \tag{2.12}$$

Then, for the sequence $Y = (1, 1)$:

$$\vec{x}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \vec{x}_1 = \begin{bmatrix} 0.25 \\ 0.75 \end{bmatrix} \quad \vec{x}_2 = \frac{\tau_1\tau_1\vec{x}_0}{\mathbb{1}^T\tau_1\tau_1\vec{x}_0} = \begin{bmatrix} 0.7 \\ 0.3 \end{bmatrix} = 0.6\vec{x}_0 + 0.4\vec{x}_1 \tag{2.13}$$

Since $\vec{x}_0$ and $\vec{x}_1$ are linearly independent, we know that such an HMM cannot have an equivalent NOOM because of NOOM's state rank constraints. In this case, we see that if an HMM reaches a state that lies strictly inside the convex hull of other reachable states, it rules out a NOOM representation. This constitutes a fairly expansive class of HMMs, suggesting that NOOMs cannot model a wide variety of HMMs, making NOOMs

a restrictive model class. The fact that NOOMs cannot represent certain HMMs, along with the fact HMMs $\subset$ PSRs, means that NOOMs cannot be equivalent to PSRs. This resolves the open question from Zhao and Jaeger [2010], and we state this corollary below:

Corollary 2.4.1 (NOOMs $\subset$ PSR)**.** *There exist finite-dimensional PSRs that have no equivalent finite-dimensional NOOMs.*

Summary of Expressiveness Relationships Thus, the expressiveness relationships we have established for NOOMs are:

1. NOOMs $\subset$ PSRs
2. HMMs $\not\subseteq$ NOOMs and NOOMs $\not\subseteq$ PSRs

Thus, NOOMs do *not* achieve our goal of finding a model class more expressive than HMMs that is also immune to the NPP. Nevertheless, they can be viewed as a precursor to a model class that does achieve this goal: hidden quantum Markov models.

2.5 Hidden Quantum Markov Models (HQMMs)

In this section, we develop the theory of hidden quantum Markov models and solve the problem we have explored in this chapter: we show that HQMMs are a learnable linear model class that avoids the negative probability problem by design while being more expressive than HMMs. There were several *a priori* reasons to look to quantum information theory: 1) the theory of open quantum systems had a probabilistic formalism to describe and update states, and 2) the state and operator representations of NOOMs bore heavy resemblance to quantum states and operations, so it was worthwhile exploring extensions of NOOMs. Happily, we show that these ideas do indeed play an important role in formulating HQMMs.

HQMMs were first proposed by Monras et al. [2010] as a class of stochastic processes for modeling the dynamics of open quantum systems. We summarize the contributions of this work as relevant to our discussion:

- **Formulate HQMMs**: This work formulates HQMMs by analogy to HMMs; they adopt quantum analogues of HMM states and operators ('density matrices' ρ and Kraus operators K_i respectively) and show how states may be updated and probabilities may be assigned to observation sequences under the standard quantum formalism.

- **Relationship between HMMs and HQMMs**: Monras et al. [2010] establish that every finite-dimensional HMM has an equivalent finite-dimensional parameterization as an HQMM by showing how to write the observable operators $\mathbf{T}_y$ of HMMs as a set of Kraus operators so that the corresponding HQMM assigns the same probabilities to observations as HMMs (showing that HMM $\subseteq$ HQMM). However, they left open the question of whether every HQMM has an equivalent representation as a finite-dimensional HMM.

- **A Compact Example**: The authors also give an example of an HQMM that can more compactly represent a given Hankel matrix than any equivalent HMM as an example of the potential advantage of HQMMs over HMMs.

However, there remained many unanswered questions; we summarize these and our direction of inquiry [Srinivasan et al., 2018b, Adhikary et al., 2019, 2020, Srinivasan et al., 2020] at a very high level below:

- **Model Compactness**: The authors' example of an HQMM that can represent a linear stochastic process in fewer dimensions than any equivalent HMM is an isolated example of the potential *compactness* of this model class. However, it does not say anything about the compactness of HQMMs in general. We find that HQMMs can in general be either more or less compact than an equivalent finite HMM (when such HMM exists), depending on the linear stochastic process being modeled. Additionally, the recipe given by Monras et al. [2010] embeds each n-dimensional observable operator $\mathbf{T}_y$ within a set of n^2 Kraus operators; we give a more efficient mapping requiring only n Kraus operators.

- **Model Expressiveness**: Separate from the question of compactness, we are interested in a notion of model expressiveness that only requires that one model class be representable *in finite dimensions* by another model class, and is agnostic to compactness. Although the work by Monras et al. [2010] contains some important elements that assist in comparing HQMM expressiveness relative to HMMs, NOOMs, and PSRs, the authors do not straightforwardly establish expressiveness relationships. We show conclusively that NOOMs $\subset$ HQMMs, HMMs $\subset$ HQMMs, and that HQMMs $\subseteq$ PSRs.

- **K-HQMMs and L-HQMMs**: We provide two different (but equivalent) formulations of HQMMs: K-HQMMs, which use Kraus operators and are identical to the formulation by Monras et al. [2010], and arise by generalizing HMMs, and L-HQMMs, which use Liouville superoperators and arise from generalizing NOOMs. These formulations are equivalent: the Liouville operators are simply a Kronecker product of Kraus operators. However, the two formulations help analyze HQMMs' relationship to HMMs and NOOMs respectively.

- **Learning HQMMs**: Monras et al. [2010] do not discuss learning HQMMs from data; we provide an iterative maximum-likelihood algorithm that can optimize the parameters of an HQMM via constrained gradient descent on the Stiefel manifold.

2.5.1 Preliminaries

Before delving deeper into our discussion on HQMMs, we present some basic background on the ideas from quantum information that will be relevant on throughout this section.

Notation $\dagger$ refers to the complex conjugate transpose. $\otimes$ refers to the Kronecker product and $\odot$ refers to the Hadamard (elementwise) product. $\mathrm{tr}_A(\rho_{AB})$ refers to the 'partial trace' of system AB over A, an operation analogous to classical marginalization that produces the quantum state of subsystem B by tracing out subsystem A. Formally [Nielsen and Chuang, 2002], given $\vec{a}_1\vec{a}_2^\dagger \otimes \vec{b}_1\vec{b}_2^\dagger \in \mathcal{H}_A \otimes \mathcal{H}_B$, with $\vec{a}_1, \vec{a}_2 \in \mathcal{H}_A$ and $\vec{b}_1, \vec{b}_2 \in \mathcal{H}_B$, we have

that $\text{tr}_B(\vec{a}_1\vec{a}_2^\dagger \otimes \vec{b}_1\vec{b}_2^\dagger) = \vec{a}_1\vec{a}_2^\dagger \text{tr}(\vec{b}_1\vec{b}_2^\dagger)$.

Pure Quantum States and the Born Rule Physically, a pure quantum state describes the *probability amplitude* of a measurable outcome in a given orthonormal basis, and the Born rule states that squaring a probability amplitude gives the *probability* of the associated measurement outcome. For our (discrete) purposes, a pure state is simply a d-dimensional column vector (denoted $|\psi\rangle$, with $\langle\psi| = |\psi\rangle^\dagger$) in a complex Hilbert space[3] $\mathcal{H}$, with the normalization condition $\||\psi\rangle\|_2^2 = 1$ given by the Born rule. In other words, each element of the pure state vector is a probability amplitude over one of d outcomes, which can be squared to get the probability of the associated outcome. For example, $|\psi\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{-i}{\sqrt{2}} \end{bmatrix}^\dagger$ is a valid quantum state, with basis states $|0\rangle$ and $|1\rangle$ having equal probability $\left\|\frac{1}{\sqrt{2}}\right\|^2 = \left\|\frac{i}{\sqrt{2}}\right\|^2 = \frac{1}{2}$. Pure quantum states live on the complex unit sphere, in contrast to discrete classical *belief states* $\vec{x} \in \mathbb{R}^d$ with $\|\vec{x}\|_1 = 1$, living on the probability simplex.

Mixed States and Density Matrices Pure quantum states capture purely quantum notions of uncertainty: squared amplitudes give a probability distribution over outcomes in a given orthonormal basis, but there is (in principle) some other orthonormal basis in which measuring the state produces a deterministic outcome. However, we may also have classical uncertainty over quantum states (called *mixed states*), where we maintain uncertainty over multiple pure states, so there is no single measurement basis in which we observe deterministic outcomes. Density matrices are a convenient representation of both pure and mixed states: given a classical probability mixture $\{p_i\}_D$ with $\sum_{i=1}^D p_i = 1$ over D quantum states $\{|\psi\rangle_i\}_D$, the density matrix is written $\rho = \sum_{i=1}^D p_i |\psi\rangle_{ii}\langle\psi|$. Density matrices are bounded operators on the underlying Hilbert space, i.e., $\rho \in L(\mathcal{H})$. For our purposes, we treat density matrices as *Hermitian positive semi-definite (PSD) matrices with unit trace*, as use them as the standard representation of quantum state. The diagonal entries in a density matrix directly give the probabilities of associated measurement outcomes. The off-

[3] A Hilbert space is a complete, inner product space.

diagonal entries contain information about measurement outcomes in other bases, but the ability to measure outcomes in different bases is usually only relevant for quantum systems. The unit trace ensures that probabilities over measurement outcomes are normalized, and the PSD-ness ensures that classical mixture probabilities are non-negative. Rank-1 density matrices correspond to pure states, while higher rank density matrices correspond to mixed states.

Unitary Matrices and Quantum Gates A unitary matrix U (satisfying $U^\dagger U = UU^\dagger = \mathbb{I}$) is the simplest valid physical evolution of a quantum state, applied as $\rho' = U\rho U^\dagger$. Unitary operations are linear, invertible isometries that preserve the unit trace normalization condition on density matrices. Quantum gates are unitary matrices that typically act on a quantum system composed of N *qubits*[4] (representable as a $2^N \times 2^N$ density matrix ρ). Unitary matrices acting on density matrices are a natural quantum analogue to column-stochastic transition operators on classical states.

Quantum Channels These are the most general valid physical evolutions of quantum states, in the sense that they include any linear map that preserves the unit trace Hermitian PSD-ness of density matrices. A quantum channel is a linear map $\mathcal{E} : L(\mathcal{H}) \to L(\mathcal{G})$ between density operators on Hilbert spaces $\mathcal{H}, \mathcal{G}$ satisfying two constraints: 1) it must be completely positive (CP), i.e., preserve the PSD-ness of input density matrices coupled with any additional arbitrary dimensional system ('ancilla'), and 2) it must be trace-preserving (TP), i.e., leave the trace of input density matrices unchanged. Physically, a quantum channel $\mathcal{E}(\cdot)$ acting on a quantum state ρ_A is essentially a local description of a unitary operation on a larger system ρ_{AB} consisting of the quantum state A and its environment B in a known state $|0_B\rangle\langle 0_B|$, i.e., $\mathcal{E}(\rho_A) = \mathrm{tr}_B\left(U(\rho_A \otimes |0_B\rangle\langle 0_B|)U^\dagger\right)$. Geometrically, quantum channels evolve density matrices on the *spectraplex* (the intersection of the convex cone of Hermitian PSD matrices with the affine space of unit trace matrices), analogous to how stochastic matrices (equivalently conditional probability tables) evolve

[4] A qubit is a quantum state with two orthogonal measurement outcomes, typically $|0\rangle$ and $1\rangle$.

classical belief states on the probability simplex.

Representations of Quantum Channels Quantum channels admit several different representations, of which we discuss the three most common ones: **Choi matrices**, **Kraus operators**, and **Liouville superoperators**. We illustrate the relationship between them in Figure 2.3.

- Let $\mathcal{H}_A$ denote the Hilbert space of pure input states with basis states $\{|\sigma\rangle\}$ and $\mathcal{H}_B$ denote the Hilbert space of pure output states with basis states $\{|\tau\rangle\}$. The **Choi matrix** is defined as $\mathbf{\Lambda}_{\mathcal{E}} = \sum_{\boldsymbol{\sigma},\boldsymbol{\tau}} |\boldsymbol{\tau}\rangle\langle\boldsymbol{\tau}'| \otimes \mathcal{E}(|\boldsymbol{\tau}\rangle\langle\boldsymbol{\tau}'|)$. We can think of the Choi matrix as having a block structure, where the input basis states $\boldsymbol{\sigma}$, $\boldsymbol{\sigma}'$ pick out a block of the Choi matrix, and output basis states $\boldsymbol{\tau}$, $\boldsymbol{\tau}'$ pick out a specific entry within the block. This representation of quantum channels is convenient when we wish to specify the constraints: the CP constraint amounts to a requirement that the Choi matrix be PSD, and the TP constraint amounts to a requirement that the partial trace $\mathrm{tr}_B(\mathbf{\Lambda}_{\boldsymbol{\mathcal{E}}}) = \mathbb{I}$ be identity. However, the Choi matrix does not provide a convenient way to compute its action on a quantum state. For these purposes, the Kraus or Liouville operator representation may be preferred.

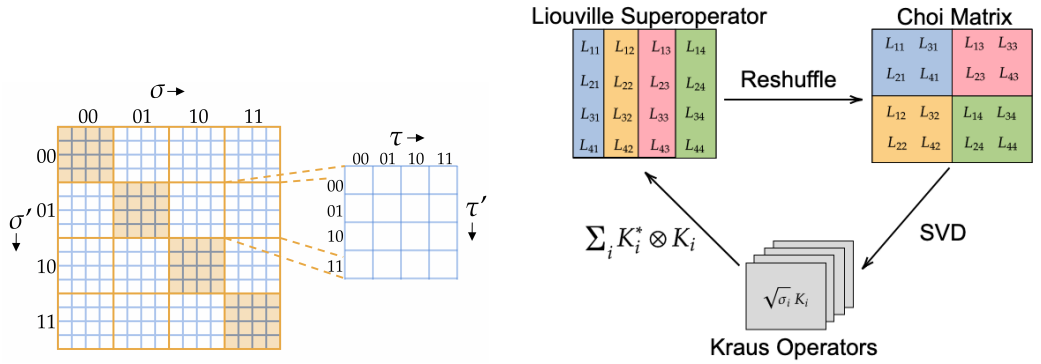
- **Kraus operators** $\{K_i\}$ are a set of matrices derived from the Schmidt decomposition (see Definition 7) of the Choi matrix satisfying the constraint $\sum_i K_i^\dagger K_i = \mathbb{I}$, allowing us to write the action of the quantum channel as $\mathcal{E}(\rho_A) = \sum_i K_i \rho K_i^\dagger$. Notice that in the case where the set consists of a single Kraus operator (i.e., the Choi matrix has unit Schmidt rank), the operator must be a unitary matrix. Additionally, as mentioned previously, a quantum channel $\mathcal{E}(\cdot)$ acting on a density matrix $\rho_A \in \mathcal{H}_A$ can equivalently be written as a unitary operation U on a larger Hilbert space $\mathcal{H}_A \otimes \mathcal{H}_B$, i.e.,

$$\mathcal{E}(\rho_A) = \sum_i K_i \rho_A K_i^\dagger = \mathrm{tr}_B \left(U(\rho_A \otimes |0_B\rangle\langle 0_B|) U^\dagger \right) \tag{2.14}$$

  We can construct this unitary U by vertically stacking the Kraus operators to get a

tall and skinny matrix κ (which must have orthonormal columns), and filling in the remaining columns with orthonormal vectors [Nielsen and Chuang, 2002].

- The **Liouville operator L** is a reshaped Choi matrix (composed by vectorizing the blocks of the Choi matrix) that acts on a *vectorized* density matrix, i.e., $\mathcal{E}(\rho_A) = \mathbf{L} \cdot \mathrm{vec}(\rho_A)$. See Figure 2.3 for illustration. We can also view the superoperator as the sum of Kronecker products of the Kraus operator's conjugate with itself: $\mathbf{L} = \sum_i \mathbf{L}_i = \sum_i K_i^* \otimes K_i$.



(a) Block structure of Choi matrix; indices (σ, σ') pick out matrix blocks and indices (τ, τ') pick out elements within the blocks.

(b) Three equivalent representations of a quantum channel

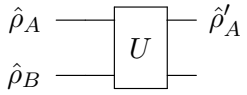
Figure 2.3: The Choi Matrix, Liouville Superoperator, and Kraus Operator Representations of Quantum Channels

We also note that the Choi matrix and Liouville superoperator representation of a quantum channel are unique, but Kraus operators are only unique up to a unitary transformation. Even the number of Kraus operators representing a quantum channel is not unique, although the Schmidt decomposition of the Choi matrix gives the canonical set of mutually orthogonal Kraus operators, and the minimum number of Kraus operators is determined by the Schmidt-rank of the Choi matrix. Quantum channels $\mathcal{E}(\cdot)$ acting on a density matrix ρ can be thought of as the quantum analogue to *applying the sum of observable operators* $\left(\sum_y \mathbf{T}_y\right)$ (which gives a column-stochastic matrix) on a classical state $\vec{x}$, amounting to 'averaging' over the observations. This hints at the fact that Kraus

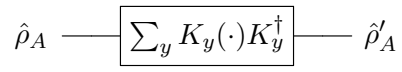
operators K_y are analogous to observable operators $\mathbf{T}_y$, which we explore shortly.

Measurement/Observation Measurement is another valid physical operation on a quantum state. The formalism of quantum measurement allows us to do two things: describe the updated quantum state after an observation, and describe the probabilities associated with various observations. In general, quantum measurements can be characterized by a set of measurement operators $\{K_y\}_{y\in\mathcal{O}}$, one for each observation $y \in \mathcal{O}$, satisfying a completeness equation $\sum_y K_y^\dagger K_y = \mathbb{I}$ to ensure that the probabilities assigned to mutually exclusive outcomes sum to 1. The probability of the quantum state ρ_X producing observation y is $P(y|x) = \mathrm{tr}(K_y \rho_X K_y^\dagger)$, while the state after measurement is $\rho_{X|y} = \frac{K_y \rho_X K_y^\dagger}{\mathrm{tr}(K_y \rho_X K_y^\dagger)}$.

Projective Measurement A particularly useful set of measurement operators are *projection operators* P_y, which project the quantum state ρ onto the basis vector of the observation. This also means that subsequent applications of the same projection operator do not change the state. Projection operators act on quantum states analogous to 'picking out the row' corresponding to an observation in the classical case. For instance, if we have a belief state $\vec{x}$, and we directly observe the system state to be in state y, using canonical basis vector $\vec{e}_y$, the probability of observation y is $P(y|x) = \vec{e}_y^T \vec{x}$, and the state after measurement is $\vec{x}_{|y} = \frac{\vec{e}_y \odot \vec{x}}{\vec{e}_y^T \vec{x}}$.



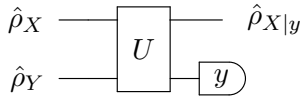
(a) State evolution as unitary operation on a larger Hilbert space followed by partial trace



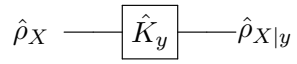
(b) State evolution as application of Kraus operators of quantum channel $\mathcal{E}(\cdot)$ on the same Hilbert space

Figure 2.4: Equivalent ways to evolve a quantum state on observation

Evolution vs Measurement with Kraus Operators Observe that measurement operators satisfy the same constraints as the Kraus operators of a quantum channel. The difference is that the state *after measurement* is updated by the Kraus operator



(a) State measurement as unitary operation on a larger Hilbert space followed by projection and partial trace



(b) State measurement as Kraus operator on the same Hilbert space, followed by re-normalization

Figure 2.5: Equivalent ways to condition a quantum state on observation

corresponding to that observation and re-normalizing ($\mathcal{E}_y(\rho) = \frac{K_y \rho K_y^\dagger}{\mathrm{tr}(K_y \rho_X K_y^\dagger)}$), while the state *after evolution* through a quantum channel is updated with all the Kraus operators $\mathcal{E}(\rho) = \sum_y K_y \rho_A K_y^\dagger$. Additionally, just as the Kraus operators of a quantum channel acting on a quantum system ρ_A can be understood as a unitary operation on a larger Hilbert space consisting of the system and its environment (Equation 2.14), the Kraus operator of a measurement acting on a quantum system ρ_A can be understood as the *same* unitary operation on the larger Hilbert space of system and environment *followed by a projective measurement* on the environment (see Figures 2.4 and 2.5):

$$K_y \rho_A K_y^\dagger = \mathrm{tr}_B \left(P_y U (\rho_A \otimes |0_B\rangle\langle 0_B|) U^\dagger P_y^\dagger \right) \tag{2.15}$$

Hence, measurement operators $\{K_y\}_{y \in \mathcal{O}}$ can equivalently be viewed as the Kraus operators of a quantum channel; when applied as in Equation 2.14, they perform state evolution, and when applied as in Equation 2.15, they condition the state on observation. This is quite analogous to the classical case where applying all observable operators on the state $\left(\sum \mathbf{T}_y\right) \vec{x}$ updates the state, while applying the observable operator corresponding to an observation and renormalizing as $\frac{\mathbf{T}_y \vec{x}}{\mathbb{1}^T \mathbf{T}_y \vec{x}}$ conditions the state on observation.

2.5.2 Formulating K-HQMMs by Generalizing HMMs

With this background, we are ready to present our formulation of K-HQMMs by generalizing HMMs from Srinivasan et al. [2018b]. The goal is to encode the same system dynamics and observation probabilities as an HMM with analogous quantum operations in finite dimensions, thereby showing HMM $\subseteq$ HQMM. This approach forestalls the fail-

Table 2.1: Summary of classical probabilistic operations and their quantum mechanical analogue

Classical probability		**Quantum Analogue**	
Description	Representation	Representation	Description
Belief State	$\vec{x}$	$\hat{\rho}$	Density Matrix
Transition Matrix	$\mathbf{A}$	$\mathcal{E}(\cdot)$	Quantum Channel
Observable Operator	$\mathbf{T}_y$	K_y	Kraus Operator
Joint Distribution	P(X, Y)	$\hat{\rho}_{XY}$	Joint Density Matrix
Marginalization	$P(X) = \sum_y P(X, Y = y)$	$\hat{\rho}_X = \mathrm{tr}_Y(\hat{\rho}_{XY})$	Partial Trace
Conditioning	$P(X\|y) = \frac{P(y,X)}{P(y)}$	$\rho_{X\|y} = \mathrm{tr}_Y(\hat{P}_y \hat{\rho}_{XY} \hat{P}_y^\dagger)$	Projection + Partial Trace

ure mode of NOOMs: developing a model that can represent probability clocks but not HMMs. Monras et al. [2010] state how to embed observable operators in Kraus operators, and this mapping uses n^2 Kraus operators for each observation; our derivation allows us to model the transition and emission matrices of HMMs with quantum channels, and efficiently represent observable operators with just n Kraus operators per observation.

Step 1: Model Transition and Emission Matrices with Unitary Matrices Given a column stochastic HMM transition matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ that acts on a quantum state $\vec{x} \in \mathbb{R}^n$, we would like to produce a unitary matrix U and quantum state ρ such that the result $A\vec{x}$ can also be extracted from $U\rho U^\dagger$. In Algorithm 1, we give such a recipe that ensures that:

$$\mathbf{A}\vec{x} = \mathrm{diag}\left(\mathrm{tr}_{\mathrm{env}}\left(U\left(\mathrm{diag}(\vec{x}) \otimes |0_{\mathrm{env}}\rangle\langle 0_{\mathrm{env}}|\right) U^\dagger\right)\right) \tag{2.16}$$

The innermost $\mathrm{diag}(\vec{x})$ creates a $n \times n$ density matrix with the entries of $\vec{x}$ on the diagonal. The outermost $\mathrm{diag}(\cdot)$ extracts the diagonal elements of the density operator in the argument. Lastly, $|0_{\mathrm{env}}\rangle\langle 0_{\mathrm{env}}|$ represents an 'ancilla' or 'environment' density matrix ρ_{env} of zeros everywhere except $\rho_{\mathrm{env}(1,1)} = 1$. This is used to enlarge the Hilbert space so we can represent the column-stochastic matrix as a unitary. We can use an identical approach to represent the HMM emission matrix $\mathbf{C}$ with a unitary matrix. Algorithm 1 gives a general recipe to model an $s \times n$ transition matrix with an $ns \times ns$ unitary matrix.

Algorithm 1 $s \times n$ Column-Stochastic Matrix to $ns \times ns$ Unitary Matrix

Input: $s \times n$ Column-Stochastic Matrix $\mathbf{A}$

Output: $ns \times ns$ block diagonal Unitary Matrix $\hat{U}$ with n blocks of $s \times s$ unitary matrices, zeros everywhere else

1: **Construct an $s \times s$ unitary matrix from each column of A**: Let c_i denote the ith column of $\mathbf{A}$. First create an $s \times s$ matrix whose each row is the square root of column c_i. Find the null space of this matrix, and you will get the $s-1$ vectors that are linearly independent of c_i. Make c_i the first column, and the remaining $s-1$ vectors the other columns of an $s \times s$ matrix.

2: **Stack each $s \times s$ matrix on a diagonal**: Follow step 1 for each column of A, and obtain n unitary matrices of dimension $s \times s$. Create a block diagonal matrix with each of these smaller unitary matrices along the diagonal, and you will obtain an $ns \times ns$ dimensional unitary matrix $\hat{U}$.

3: **Note:** The unitary operator constructed here is designed to be applied on a density matrix tensored with an environment density matrix prepared with zeros everywhere except $\hat{\rho}_{1,1} = 1$.

Step 2: Setup a 'Quantum Circuit' to Implement Transition and Emission

Next, we construct a 'quantum circuit' diagram showing how these unitary matrices corresponding to transition and emission matrices can be applied on the quantum state $\rho_{X_{t-1}} \in \mathcal{L}(\mathcal{H}_X)$[5] to generate the same statistics as the transition and emission matrices in Figure 2.6. We begin by preparing the 'ancilla' or 'environment' states $\hat{\rho}_{X'_t} \in \mathcal{L}(\mathcal{H}_{X'})$ and $\hat{\rho}_{Y_t} \in \mathcal{L}(\mathcal{H}_Y)$ appropriately (i.e., entirely in system state 1, represented by a density matrix of zeros except $\hat{\rho}_{1,1} = 1$), and using Algorithm 1 to construct $\hat{U}_1$ and $\hat{U}_2$ from transition matrix $\mathbf{A}$ and emission matrix $\mathbf{C}$ respectively. $\hat{U}_1$ evolves $\left(\hat{\rho}_{X_{t-1}} \otimes \hat{\rho}_{X'_t}\right)$ to perform Markovian transition to produce $\hat{\rho}'_{X_t}$[6], reproducing the action of the transition matrix $\mathbf{A}$ on the classical belief state $\vec{x}$ (as in Equation 2.1):

$$\hat{\rho}'_{X_t} = \mathrm{tr}_X(\hat{U}_1(\hat{\rho}_{X_{t-1}} \otimes \hat{\rho}_{X'_t})\hat{U}_1^\dagger) \tag{2.17}$$

$\hat{U}_2$ then evolves $(\hat{\rho}'_{X_t} \otimes \hat{\rho}_{Y_t})$ to prepare the joint distribution of latent state and observation $\hat{\rho}_{X_tY_t}$; the diagonal entries of $\hat{\rho}_{X_tY_t} \in \mathbb{R}^{sn \times sn}$ encode the $s \times n$ joint probability

[5] We use $\mathcal{L}(\mathcal{H})$ to refer to the space of bounded linear operators on Hilbert space $\mathcal{H}$.

[6] The latent state $\hat{\rho}'_{X_t}$ is transferred from its original Hilbert space $\mathcal{L}(\mathcal{H}_X)$ (where $\hat{\rho}_{X_{t-1}}$ lived) to the Hilbert space $\mathcal{L}(\mathcal{H}_{X'})$ of $\hat{\rho}_{X'_t}$. This is done to keep the conversion of stochastic matrix to unitary matrix in Algorithm 1 consistent for both transition and emission matrices.

table of latent state and observations. Then, when we make an observation, we treat it as a measurement on the second subsystem (Hilbert space $\mathcal{L}(\mathcal{H}_Y)$) which is equivalent to applying projection operators on joint density matrix $\hat{\rho}_{X_t Y_t}$ (i.e., projecting onto the basis vectors consistent with the measurement) and renormalizing. Subsequently, partial trace over the second subsystem (Hilbert space of $\mathcal{L}(\mathcal{H}_Y)$) gives the latent state conditioned on the observation $\hat{\rho}_{X_t|y_t}$.

$$\hat{\rho}_{X_t|y_t} = \frac{\mathrm{tr}_Y(P_{y_t}\hat{U}_2(\hat{\rho}'_{X_t} \otimes \hat{\rho}_{Y_t})\hat{U}_2^\dagger P_{y_t}^\dagger)}{\mathrm{tr}(P_{y_t}\hat{U}_2(\hat{\rho}'_{X_t} \otimes \hat{\rho}_{Y_t})\hat{U}_2^\dagger P_{y_t}^\dagger)} \qquad P(y|X) = \mathrm{tr}(P_{y_t}\hat{U}_2(\hat{\rho}'_{X_t} \otimes \hat{\rho}_{Y_t})\hat{U}_2^\dagger P_{y_t}^\dagger) \quad (2.18)$$

Thus, the full state update expression (corresponding to Figure 2.6) that explicitly models a hidden Markov Model on a quantum circuit can be written as:

$$\hat{\rho}_{X_t|y_t} \propto \mathrm{tr}_{\mathcal{H}_Y}\left(\hat{P}_{y_t}\hat{U}_2\left(\mathrm{tr}_{\mathcal{H}_X}\left(\hat{U}_1(\hat{\rho}_{X_{t-1}} \otimes \hat{\rho}_{X'_t})\hat{U}_1^\dagger\right) \otimes \hat{\rho}_{Y_t}\right)\hat{U}_2^\dagger \hat{P}_y^\dagger\right) \quad (2.19)$$

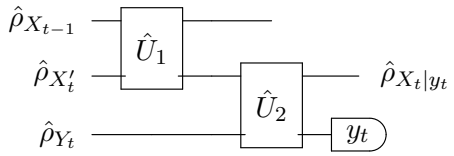


Figure 2.6: A 'quantum circuit' that generates the same statistics as an HMM; $\hat{U}_1$ and $\hat{U}_2$ encode the information in HMM transition and emission matrices **A** and **C** respectively, using Algorithm 1

.

Step 3: Simplify from Unitary Matrices to Kraus Operators As we discussed in Section 2.5.1, unitary operators acting on the joint Hilbert space of a system and environment can be reduced to Kraus operators acting on system subspace, when the environment is always prepared in the same way. Thus, we can simplify the circuit in Figure 2.6 (corresponding to expression in Equation 2.19) to use Kraus operators acting on the lower-dimensional state space of $\hat{\rho}_{X_t}$ using the equivalence shown in Figures 2.4 and 2.5. We won't give the recipe for converting the unitaries to their Kraus operator representation (see Srinivasan et al. [2018b] for details), but we will state how to convert

observable operators into Kraus operators shortly.

Encoding $\hat{U}_1$ acting on $\mathcal{L}(\mathcal{H}_X)\otimes\mathcal{L}(\mathcal{H}_{X'})$ with the environment fixed to $\hat{\rho}_{X'_t} = |0_{X'}\rangle\langle 0_{X'}|$ as a set of Kraus operators $\{K_w\}$ representing a quantum channel $\mathcal{K} : \mathcal{L}(\mathcal{H}_X) \to \mathcal{L}(\mathcal{H}_{X'})$ (with the operators satisfying $\sum_w \hat{K}_w^\dagger \hat{K}_w = \mathbb{I}$) simplifies Equation 2.17 to:

$$\hat{\rho}'_{X_t} = \mathcal{K}(\hat{\rho}_{X_{t-1}}) = \sum_w K_w \hat{\rho}_{X_{t-1}} K_w^\dagger \tag{2.20}$$

The number of Kraus operators $|w|$ in the set $\{K_w\}$ is determined by $\dim(\mathcal{H}_{X'})$, which in this case is the same as $\dim(\mathcal{H}_X)$, so $|w| = n$. Similarly, we can encode $\hat{U}_2$ acting on $\mathcal{L}(\mathcal{H}_{X'})\otimes\mathcal{L}(\mathcal{H}_Y)$ with the environment fixed to $\hat{\rho}_{Y_t} = |0_Y\rangle\langle 0_Y|$ as a set of Kraus operators $\{K_y\}$ representing a quantum channel $\mathcal{K} : \mathcal{H}_{X'} \to \mathcal{H}_Y$ (with the operators satisfying $\sum_y \hat{K}_y^\dagger \hat{K}_y = \mathbb{I}$). Note that the the Kraus operators $\{K_w\}$ encoding $\hat{U}_1$ implement a quantum channel analogous to Markovian transition, hence all the Kraus operators are applied. Here, the Kraus operators $\{K_y\}$ encoding $\hat{U}_2$ are used as measurement operators for conditioning on observation, so we only apply the Kraus operator corresponding to our observation[7]. This simplifies Equation 2.18 (shown graphically in Figure 2.7a) to:

$$\hat{\rho}_{X_t|y_t} = \frac{K_y \hat{\rho}'_{X_t} K_y^\dagger}{\mathrm{tr}\left(K_y \hat{\rho}_{X'_t} K_y^\dagger\right)} \qquad P(y|X) = \mathrm{tr}\left(K_y \hat{\rho}_{X'_t} K_y^\dagger\right) \tag{2.21}$$

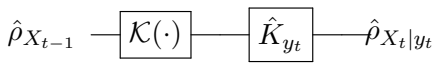
Step 4: Collect the Unitaries into Single Set of Kraus Operators A final simplification is to combine the set of s Kraus operators $\{K_w\}$ of the quantum channel (analogous to transition matrices) with the Kraus operators of measurement $\{K_y\}$ (analogous to emission matrices), to obtain a single set of Kraus operators (analogous to observable operators). Right-multiplying each Kraus operator in $\{\hat{K}_w\}$ with each operator in $\{\hat{K}_y\}$ as $K_{w,y} = K_y K_w$, we have a set of Kraus operators $\{\hat{K}_{w_y,y}\}$. Note that each observation y now has a *set* of associated Kraus operators $\{K_{w_y,y}\}$ that must be applied in order to simulate transition followed by observation, as opposed to the single-operator-per-measurement

[7]Indeed, we can expect there to be a single Kraus operator for each observation, although these Kraus operators need not be orthogonal. We can also interpret the number of Kraus operators (either in a quantum channel or set of measurement operators) as the dimension of the larger system within which the they have a unitary representation.

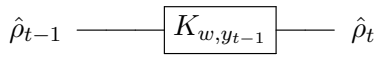
$\{K_y\}$ previously. Furthermore, there are exactly n Kraus operators $\{K_{w_y,y}\}$ per observation y, a more efficient representation than the n^2 Kraus operators per observation given by Monras et al. [2010]. Finally, we can simulate HMMs with quantum systems by using the derived Kraus operators $\{K_{w_y,y}\}$ to update state and compute observation probabilities as follows:

$$\hat{\rho}_{X_t} = \frac{\sum_{w_y} \hat{K}_{w_y,y_t} \hat{\rho}_{X_{t-1}} \hat{K}^{\dagger}_{w_y,y_t}}{\mathrm{tr}\left(\sum_{w_y} \hat{K}_{w_y,y_t} \hat{\rho}_{X_{t-1}} \hat{K}^{\dagger}_{w_y,y_t}\right)} \tag{2.22}$$

Setting Kraus Operators Directly from Observable Operators The procedure outlined above allows us to mimic HMM state evolution and observation probabilities. However, we did not specify how to go from the unitaries U_1 and U_2 to the Kraus operators $\{K_w\}$ and $\{K_y\}$ and thus to $\{K_{w_y,y}\}$. We show how this can be done in Srinivasan et al. [2018b], although practically we needn't construct the final Kraus operators $\{K_{w_y,y}\}$ through this route; an equivalent but simpler approach is to directly use observable operators $\{\mathbf{T}_y\}$, and set the wth column of $\hat{K}^{(:,w)}_{w_y,y} = \sqrt{\mathbf{T}^{(:,w)}_y}$, with all other entries being zero. This ensures $\sum_{w_y,y} \hat{K}^{\dagger}_{w_y,y} K_{w_y,y} = \mathbb{I}$.



(a) Simplification of the unitaries in Figure 2.6 into a quantum channel $\mathcal{K}(\cdot)$ with Kraus operators $\{K_w\}$ and measurement with Kraus operators $\{K_y\}$



(b) Further simplification of Figure 2.7a by combining the quantum channel and measurement operators into a single set of Kraus operators $\{K_{w_y,y}\}$

Figure 2.7: Simplified scheme to generate the same statistics as HMMs where the unitary matrices have been condensed to Kraus operators

Formulating K-HQMMs When the operators $\{K_{w_y,y}\}$ are constructed according to the recipe above, the model can simulate HMMs, but if we let these operators be *any* valid Kraus operators and define them over the complex field, we have a general model we refer to as 'hidden quantum Markov models'. Additionally, note that the number of Kraus operators per observation $|w|$ is determined by the dimension of the ancilla/environment needed to implement transition dynamics. For HQMMs to implement HMMs, we must

have $|w| = n$ (n is the dimension of the latent state). At the other extreme, if $|w| = 1$, (i.e., only a single Kraus operators K_y per observation) the latent state dynamics are modeled as unitary and we have a complex-valued NOOM. We give our formal definition of K-HQMMs here [Adhikary et al., 2019]:

Definition 8 (K-HQMMs)**.** *An n-dimensional Kraus-Hidden Quantum Markov Model with a set of discrete observations $\mathcal{O}$ is a tuple $(\mathbb{C}^{n\times n}, tr(\cdot), \{K_{y,w_y}\}_{y\in\mathcal{O}}, \rho_0)$ where initial state $\rho_0 \in \mathbb{C}^{n\times n}$ and Kraus operators $\{K_{y,w_y}\}_{y\in\mathcal{O},w_y\in\mathbb{N}} \in \mathbb{C}^{n\times n}$ satisfy the following constraints:*

1. ***Density Matrix as State Representation**: ρ_0 is a Hermitian PSD matrix of arbitrary rank,*
2. ***Normalized Initial State**: $tr(\rho_0) = 1$,*
3. ***Normalized marginal over observations ('Trace-Preservation')**: $\sum_{y,w} K_{y,w}^\dagger K_{y,w} = \mathbb{I}$.*

The state update after observing y is computed according to Equation 2.22 and probability of a given sequence is given by:

$$P(y_1, \ldots, y_T) = \text{tr}\left(\sum_{w_{y_T}} K_{y_T,w_{y_T}} \ldots \left(\sum_{w_{y_1}} K_{y_1,w_{y_1}} \rho_0 K_{y_1,w_{y_1}}^\dagger\right) \ldots K_{y_T,w_{y_T}}^\dagger\right) \tag{2.23}$$

Now, we can use the correspondence between Kraus operators and Liouville operators described in Section 2.5.1 and define HQMMs via the Liouville representation (as [Monras et al., 2010] do). However, we develop that formulation by generalizing NOOMs (analogous to how we K-HQMMs by generalizing HMMs), to show that NOOMs $\subseteq$ HQMMs. This approach also allows to connect HQMMs to PSRs.

2.5.3 Formulating L-HQMMs by Generalizing NOOMs

We will show how HQMMs can be defined by generalizing NOOMs [Adhikary et al., 2019], in the sense that we propose a parameterization of HQMMs that includes NOOMs as a special case. We do so by allowing parameters to be complex and generalizing NOOM states and operators from the representation from Definition 6.

Generalizing NOOM States We have the tools from Section 2.5.1 to recognize that the NOOM initial state $\vec{\rho}_0 = \vec{\psi}_0 \otimes \vec{\psi}_0$ in the PSR representation (Equation 2.10) can be viewed as a vectorized rank-1 unit trace Hermitian matrix since $\|\vec{\psi}\|_2 = 1$:

$$\vec{\rho}_0 = \vec{\psi}_0 \otimes \vec{\psi}_0 = \text{vec}\left(\vec{\psi}_0 \vec{\psi}_0^\dagger\right) \tag{2.24}$$

$\vec{\psi}_0 \vec{\psi}_0^\dagger$ is clearly a rank-1 Hermitian PSD matrix, and $\text{tr}(\vec{\psi}_0 \vec{\psi}_0^\dagger) = \text{tr}(\vec{\psi}_0^\dagger \vec{\psi}_0) = \|\vec{\psi}_0\|_2^2 = 1$. A natural step would be to let the initial state be vectorized matrices of arbitrary rank, i.e., $\hat{\rho}_0 = \sum_i p_i \vec{\psi}_i \vec{\psi}_i^\dagger$ instead. The normalization condition on the initial state can then be restated as $1 = \sigma^\dagger \vec{\rho}_0 = \vec{\mathbb{I}}_{n^2}^T \vec{\rho}_0 = \text{tr}\left(\hat{\rho}_0\right) = \sum_i p_i$. As a linear combination of outer products, $\hat{\rho}$ remains Hermitian. If we impose no further constraints, we could allow p_i to be complex-valued or negative as long as the normalization condition above was satisfied. However, this could once again lead to negative probabilities when applying σ to an evolved state, and hence a non-constructive model. Thus, we impose a positive semi-definiteness (PSD) constraint on the initial state to guarantee that $p_i \in \mathbb{R}_{\geq 0}$ so that $\text{tr}(\hat{\rho}_0)$ is real and non-negative. Essentially, the NOOM initial states were *vectorized rank-1 density matrices* $\vec{\rho}_0$ (or from a quantum mechanical perspective, 'pure states'), and we are now proposing a model whose initial states $\vec{\rho}_0$ are vectorized arbitrary-rank density matrices.

Generalizing NOOM operators Having defined a convex cone of valid states (Hermitian PSD matrices), we now generalize NOOM operators to ensure that states always evolve inside the cone. Once again from Equation 2.10, we know that the NOOM operators in the PSR representation are Schmidt-rank 1 operators $\tau_y = \phi_y \otimes \phi_y$ satisfying

$\vec{\mathbb{I}}^T \left(\sum_y \tau_y\right) = \vec{\mathbb{I}}^T$. Here, the natural step would be to allow the NOOM operators to be *arbitrary Schmidt rank* operators that satisfy the same constraint. Doing so yields the Liouville superoperator representation $\{\mathbf{L}_y\}_{y\in\mathcal{O}}$ of quantum channels. From Section 2.5.1 we know quantum channels satisfying CP and TP constraints are sufficient to ensure that density matrices evolve inside the spectraplex. The TP constraint on superoperators is ($\vec{\mathbb{I}}^T \left(\sum_y \mathbf{L}_y\right) = \mathbb{I}^T$), identical to the NOOM normalization constraint. The additional constraint on superoperators must satisfy is the complete positivity (CP) constraint, ensuring that they always preserve the Hermitian PSD condition of the state, even when combined with a larger system. While we can't neatly describe these constraints on the Liouville operator, we can do so on the Choi matrix, 'reshuffled' version of the superoperator [Wood et al., 2015, Życzkowski and Bengtsson, 2004]. We illustrate this relationship in Figure 2.3.

Thus, taking the rank-1 vectorized density matrix of NOOMs to arbitrary rank vectorized density matrices and the Schmidt-rank 1 operators of NOOMs to arbitrary Schmidt-rank operators produces L-HQMMs and makes it clear that a NOOMs are a special case of HQMMs, and use the *exact same parameterization*; they do not need to be transformed or embedded in the way that HMMs operators needed to be transformed to HQMM operators. This procedure also gives L-HQMMs in a PSR-like representation with evaluation functional $\vec{\mathbb{I}}$, operators $\{\mathbf{L}_y\}$ and initial state $\vec{\rho}_0$ where these parameters have the previously discussed structure, clearly showing that HQMMs are a special case of PSRs. We can now give a formal definition of L-HQMMs:

Definition 9 (L-HQMMs)**.** *An n^2-dimensional Liouville-Hidden Quantum Markov Model with a set of discrete observations $\mathcal{O}$ is a tuple $(\mathbb{C}^{n^2}, \vec{\mathbb{I}}, (\mathbf{L}_y)_{y\in\mathcal{O}}, \vec{\rho}_0)$ where the initial state $\vec{\rho}_0 \in \mathbb{C}^{n^2}$ and Liouville superoperators $\{\mathbf{L}_y\}_{y\in\mathcal{O}} \in \mathbb{C}^{n^2\times n^2}$ with corresponding Choi matrices $\{\mathbf{C}_y\}_{y\in\mathcal{O}}$ satisfy the following constraints:*

1. ***Vectorized Density Matrix as State Representation**: $\vec{\rho}_0$ is a vectorized Hermitian PSD matrix of arbitrary rank,*
2. ***Normalized initial state**: $\vec{\mathbb{I}}^T\vec{\rho}_0 = 1$,*

3. ***Complete Positivity***: $\mathbf{C}_y \geq 0$ *(Choi matrix is PSD).*

4. ***Trace-Preservation***: $\vec{\mathbb{I}}^T \left(\sum_{y \in \mathcal{O}} \mathbf{L}_y \right) = \vec{\mathbb{I}}^T$

For such a model, the state update after observing $y \in \mathcal{O}$ and computing the probability of that observation are:

$$\vec{\rho}_t = \frac{\mathbf{L}_y \vec{\rho}_{t-1}}{\vec{\mathbb{I}}^T \mathbf{L}_y \vec{\rho}_{t-1}} \qquad\qquad P(\bar{y}) = \vec{\mathbb{I}}^T \mathbf{L}_{y_t} \ \ldots \ \mathbf{L}_{y_1} \vec{\rho} \tag{2.25}$$

Lastly, we note that L-HQMMs are exactly equivalent to K-HQMMs in the sense that L-HQMMs are a 'vectorized' version of K-HQMMs: the L-HQMM state $\vec{\rho}$ is a vectorization of the K-HQMM state ρ_0, the L-HQMM operators $\mathbf{L}_y = \sum_{w_y} K^*_{w_y,y} \otimes K_{w_y,y}$, and the L-HQMM evaluation functional $\vec{\mathbb{I}}$ acting on a vectorized density matrix $\vec{\rho}$ is equivalent to taking the trace of the density matrix itself. The L-HQMM representation is typically useful for fast computation (while being more memory intensive), while the K-HQMM representation is more helpful for crafting learning algorithms.

2.5.4 HQMMs vs {NOOMs, PSRs, HMMs}

When HQMMs were first proposed, Monras et al. [2010] established that HMMs $\subseteq$ HQMMs, but left the question of whether HMMs = HQMMs open. Additionally, the relationship with NOOMs and PSRs was not explored. In light of the discussion thus far, we can establish the following results:

Theorem 5 (HMMs $\subset$ HQMMs and NOOMs $\subset$ HQMMs)**.** *There exist finite-dimensional HQMMs that do not have any equivalent representation as a finite-dimensional HMM. Also, there exist finite-dimensional HQMMs that do not have any equivalent representation as a finite-dimensional NOOM.*

Proof. Section 2.5.2 established that HMMs $\subseteq$ HQMMs, and Section 2.5.3 established that NOOMs $\subseteq$ HQMMs. The NOOM probability clock (discussed in Section 2.4.1) established that NOOMs $\not\subseteq$ HMMs, and Theorem 4 established that HMM $\not\subseteq$ NOOMs. Thus, we can conclude that HMMs $\subset$ HQMMs and NOOMs $\subset$ HQMMs. $\square$

Theorem 6. *HQMMs $\subseteq$ PSRs, and the pointed convex cone of valid initial states for HQMMs is a spectraplex.*

Proof. This is easy to see with the L-HQMM formulation. The valid initial states of the L-HQMM are vectorized unit trace Hermitian PSD matrices (density matrices) $\vec{\rho}_0$, and $\vec{\mathbb{I}}\vec{\rho}_0 = \mathrm{tr}(\rho_0) = 1$ satisfying condition 1 of PSR Definition 5 by construction. Condition 2 is satisfied by the TP constraint on HQMM operators $\mathbf{L}_y$, which assures $\vec{\mathbb{I}}^T \sum_y \mathbf{L}_y = \vec{\mathbb{I}}^T \mathbf{L} = \vec{\mathbb{I}}^T$. Finallly, CP constraints on L-HQMM operators guarantee that $\mathbf{L}_{\bar{y}}\vec{\rho}$ always yields a vectorized Hermitian PSD matrix, so the trace of this matrix is always real and non-negative, i.e., $P(\overline{y}) = \vec{\mathbb{I}}^T \mathbf{L}_{\bar{y}}\vec{\rho} \geq 0$. This satisfies condition 3 of the PSR definition, and we see that HQMMs $\subseteq$ PSRs.

The valid initial states of L-HQMMs are Hermitian PSD matrices with unit trace. As discussed in Section 2.5.1, Hermitian PSD matrices form a convex cone, and the intersection of this cone with the linear affine subspace of trace 1 matrices is a spectrahedron known as a spectraplex. □

2.5.5 The Completely Positive Realization Problem: Are HQMMs = PSRs?

The relationships we have established thus far are visualized in Figure 2.8. We have answered the question we set out to answer in this chapter: HQMMs are a more expressive model class compared to HMMs, and HQMMs avoid the NPP by design (we discuss learning HQMMs in Section 2.5.6). As illustrated, the big question that remains unanswered is whether there are any finite-dimensional PSRs that do not have any equivalent representation as a finite-dimensional HQMM. The canonical example of PSRs that had no finite-dimensional HMMs was the probability clock, as HMMs could not capture oscillating probabilities and only admitted polyhedral convex cones of initial states, as discussed in Section 2.3.3. However, HQMMs *can* capture the probability clock, and admit infinitely generated convex cones of initial states, so there is currently no known linear stochastic process that cannot be represented by an HQMM. This also suggests that HQMMs largely

capture the important way in which PSRs are more expressive than HMMs. Nevertheless, a formal proof answering the question of whether PSRs are *strictly* more expressive than HQMMs (i.e., of an expressiveness 'gap' between PSRs and HQMMs) is yet to be found, although we make some remarks and discuss previous work in the literature.

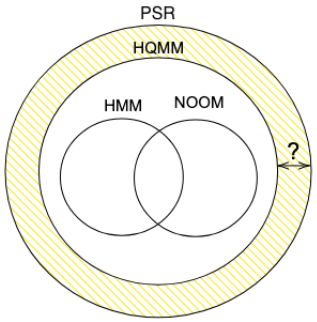


Figure 2.8: Established expressiveness relationships between PSRs, HQMMs, NOOMs, and HMMs

First, we note that the undecidability of the negative probability problem precludes an algorithm that can find an HQMM representation for a candidate PSR if such representation existed; if such an algorithm existed, we could apply it on a candidate set of parameters and check if the result was a valid HQMM. However, an algorithm that doesn't solve the decision problem may still be possible; for instance, an algorithm that maps both valid and invalid PSRs (i.e., those that satisfy conditions 1 and 2 of the PSR definition) to valid HQMMs may be possible.

One might wonder whether we can construct the HQMM representing a PSR using the strategy we used to construct the HQMM representation of an HMM in Section 2.5.2; after all, HQMMs admit negative and complex entries, so taking the square root of the elements of the observable operator is not a concern. Unfortunately, this is not a viable approach. If we set $\hat{K}^{(:,w)}_{w_y,y} = \sqrt{\mathbf{T}^{(:,w)}_y}$, we find $\hat{K}^{(:,w)\dagger}_{w_y,y}\hat{K}^{(:,w)}_{w_y,y} \neq \mathbf{T}_y$ in general because the complex entries in $K_{w_y,y}$ are conjugated in $K^{\dagger}_{w_y,y}$, so operators constructed this way do not satisfy the Kraus operator requirement of $\sum_{w_y,y} K^{\dagger}_{w_y,y}K_{w_y,y} = \mathbb{I}$. This is not an issue in the case that $\mathbf{T}_y$ is non-negative, since the square-roots are real and conjugation does not have any effect.

Completely Positive Realization Problem In an analogy to the positive realization problem [Vidyasagar, 2011] (the question of which PSRs have equivalent HMM representations), Monras and Winter [2016] term the question of which PSRs have equivalent HQMMs as the *completely positive realization problem* (CPRP). They argue that a necessary and sufficient condition for a PSR to have an equivalent HQMM is if the PSR operators come from a semi-definite representable (SDR) cone that also defines the convex cone of valid initial states.[8] They suggest that we could show that HQMMs are equivalent to PSRs if either we could show that every PSR's convex cone satisfied the SDR condition, or if we could show that every PSR's convex cone was a semi-algebraic set and appeal to the Helton-Nie conjecture. The Helton-Nie conjecture suggested that every convex and semi-algebraic set was SDR. While it is the case that every SDR set is convex and semi-algebraic, the converse (i.e., the Helton-Nie conjecture) was later shown to be false [Scheiderer, 2018]. Still, convex semi-algebraic sets are SDR sets in a wide variety of cases, so HQMMs may still be equivalent to PSRs. We thus pose two open questions that must be answered to obtain a full characterization of the expressiveness of HQMMs relative to PSRs: first, whether the convex cone characterizing the PSR is semi-algebraic, and second, if it satisfies any sufficient conditions for being SDR [Helton and Nie, 2009], and if not, how common such conditions are. Nevertheless, convex and semi-algebraic cones that are SDR are a broad class and our results thus far show that HQMMs may be the most expressive known subset of PSRs. Indeeed, we do not know any examples of PSRs that do not have an equivalent finite-dimensional HQMM. Our conjecture, based on the fact that HQMMs can capture infinitely generated cones, is that HQMMs are equivalent to PSRs. Settling this claim would be a valuable direction for future work.

2.5.6 A Learning Algorithm for HQMMs

Having formulated HQMMs, we turn to the question of how HQMMs can be learned from data. EM is the standard approach to learning HMMs, but as previously discussed,

[8] Linear maps of spectrahedra are called *spectrahedral shadows* or semi-definitely representable sets. These are the feasible regions of a semidefinite program.

HMMs are limited in their expressiveness. PSRs can be learned via a spectral algorithm, but it is undecidable whether learned model parameters are valid. To combine the greater expressiveness of PSRs with the learnability of HMMs, we need a learning algorithm for HQMMs. When Monras et al. [2010] formulated HQMMs, they did not provide a learning algorithm. Schuld et al. [2015] proposed identifying a learning algorithm for HQMMs as an important direction for research, and we present our work in Srinivasan et al. [2018b], Adhikary et al. [2019], and Srinivasan et al. [2020] in developing a method to learn HQMMs from data.

The Loss Function Since HQMMs are fundamentally probabilistic models, the negative log-likelihood of the data is a natural choice of loss function. This yields the following log likelihood of a sequence $y_1, \ldots, y_T$ for a K-HQMM parameterized by the set of Kraus operators $\{\hat{K}_{y,w}\}$:

$$\mathcal{L} = -\ln\ \mathrm{tr}\left(\sum_w K_{y_T,w} \ldots \left(\sum_w K_{y_1,w}\rho_0\ K_{y_1,w}^\dagger\right) \ldots K_{y_T,w}^\dagger\right) \tag{2.26}$$

In the L-HQMM formulation with parameters $\{\mathbf{L}_y\}$, the log likelihood can be written as:

$$\mathcal{L} = -\ln\left(\vec{\mathbb{I}}^T \mathbf{L}_{y_T} \ldots \mathbf{L}_{y_1} \vec{\rho}_0\right) \tag{2.27}$$

Note that we cannot simply take the gradient[9] of the loss $\mathcal{L}$ with respect to the parameters and step along it, the learned Kraus operators must satisfy the TP constraint (condition 3 in Definition 8) $\sum_{y,w} K_{y,w}^\dagger K_{y,w} = \mathbb{I}$. The problem of learning a set of N trace-preserving $n \times n$ Kraus operators can equivalently be framed as one of learning a matrix $\boldsymbol{\kappa} \in \mathbb{C}^{nN \times n}$ satisfying $\boldsymbol{\kappa}^\dagger \boldsymbol{\kappa} = \mathbb{I}$, where $\boldsymbol{\kappa}$ can be block-partitioned row-wise into the N Kraus operators that parameterize the HQMM. A natural approach is to begin with an initial guess $\boldsymbol{\kappa}_0$ with a pre-determined partitioning into the Kraus operators we

[9]Since $\mathcal{L}$ is a function of complex matrices, the direction of steepest descent corresponds to the gradient with respect to the complex conjugate of the Kraus operators [A.Hjørungnes and D.Gesbert, 2007].

wish to learn, setup a computational graph according to Equation 2.26, and compute and backpropagate gradients of the log-likelihood to update $\boldsymbol{\kappa}$. However, we must still specify how we carry out this update while preserving the TP constraint.

A First Attempt: Givens Rotations Since $\boldsymbol{\kappa}$ is a matrix with orthonormal columns, any initial guess $\boldsymbol{\kappa}_0$ is a unitary transformation away from the true $\boldsymbol{\kappa}^*$ that maximizes the log-likelihood. Our first attempt [Srinivasan et al., 2018b] iteratively finds a series of Givens rotations that locally increase the log-likelihood, generalizing a learning algorithm for NOOMs proposed by Zhao and Jaeger [2007]. However, a Givens rotation only changes two rows of $\boldsymbol{\kappa}$ at a time, making this approach prohibitively slow for learning large $\boldsymbol{\kappa}$ matrices. Furthermore, since these two rows are picked at random, this approach is not guaranteed to step towards the optimum at every iteration. In Section 2.5.7, we discuss the empirical performance of this approach.

A Better Approach: Optimization on the Stiefel Manifold In Adhikary et al. [2019, 2020], we identify that the the TP constraint on Kraus operators is *exactly* the requirement that $\boldsymbol{\kappa}$ lies on the Stiefel manifold[10], enabling us to use any of the many existing approaches to optimization on the Stiefel manifold. There are a number of algorithms for constrained optimization on the Stiefel manifold, and they usually are either projection-like (which re-orthogonalize the naive gradient descent updates) or geodesic-like (which directly generate updates on the manifold itself) [Jiang and Dai, 2013]. We picked the geodesic like algorithm proposed by Wen and Yin [2013a] for its ease of implementation and its performance in our experiments (see Appendix A.1.1 for details).

The idea is that given a gradient G of the loss function $\mathcal{L}$ with respect to parameters $\boldsymbol{\kappa}$, we would like to move along a trajectory $\gamma(\tau)$ for some step size τ, such that moving along this curve corresponds to stepping along the direction of the gradient while staying on the manifold. We can achieve this through *retractions*, which smoothly map the tangent bundle of the manifold onto the manifold itself, while preserving the direction of the

[10] The Stiefel manifold $\mathcal{V}_k(\mathbb{C}^n) = \{\kappa \in \mathbb{C}^{n\times k} | \boldsymbol{\kappa}^\dagger \boldsymbol{\kappa} = \mathbb{I}\}$.

gradient at that point [Absil et al., 2007]. The gradient G of $\mathcal{L}$ corresponds to a vector in the tangent bundle of this manifold, so intuitively we can imagine a retraction as wrapping the direction of G onto the surface of the manifold, providing a feasible path for curvilinear descent. Note that since $\mathcal{L}$ is a function of complex matrices, the direction of steepest descent corresponds to the *gradient with respect to the complex conjugate* of the Kraus operators [A.Hjørungnes and D.Gesbert, 2007].

Wen and Yin [2013a] present the following Crank-Nicolson-like update $\gamma(\tau)$ on the Stiefel manifold with respect to an initial feasible solution κ_0:

$$\gamma(\tau) = \left(\mathbb{I} + \frac{\tau}{2}A\right)^{-1}\left(\mathbb{I} - \frac{\tau}{2}A\right)\kappa_0, \tag{2.28}$$

where A is a skew-symmetric matrix defined as:

$$A = G\kappa_0^\dagger - \kappa_0 G^\dagger \tag{2.29}$$

If the path $\gamma(\tau)$ is the direction of steepest descent to feasibly optimize Equation 2.26, it must fulfill two criteria. First, when we take no steps along the path (i.e., when $\tau = 0$), we should stay at the initial point κ_0 on the manifold, and the curve $\gamma(\tau)$ should point in the direction of the gradient with respect to $\mathcal{L}$. This can be easily verified as $\gamma(0) = \kappa_0$ and $\gamma'(0) = -G$ [Wen and Yin, 2013a]. Second, moving along $\gamma(\tau)$ should not move us off the manifold regardless of the step-size τ. Since the path $\gamma(\tau)$ in Equation 2.28 is the Cayley transform of the skew-symmetric matrix A applied to κ_0, we know $\gamma(\tau)^\dagger\gamma(\tau) = \kappa_0^\dagger\kappa_0 = \mathbb{I}$. This ensures that as long as we begin with a point on the Stiefel manifold, the update in Equation 2.28 will keep the point on the manifold for any τ and G.

In practice, the matrix $\kappa \in \mathbb{C}^{nN\times n}$ is 'tall and skinny' and we would like to avoid computing the inverse $\left(\mathbb{I} + \frac{\tau}{2}A\right)^{-1}$ where $A \in \mathbb{C}^{nN\times nN}$. Hence, we use the following equivalent but computationally more efficient expression for $\gamma(\tau)$ recommended by Wen and Yin [2013a]:

$$\gamma(\tau) = \kappa_0 - \tau U\left(\mathbb{I} + \frac{\tau}{2}V^\dagger U\right)^{-1}V^\dagger\kappa_0, \tag{2.30}$$

where $U = [G \mid \kappa_0]$ and $V = [\kappa_0 \mid -G]$. This only requires inverting a $2n \times 2n$ matrix.

Importantly, this method can be combined with a gradient descent scheme (summarized in Algorithm 2) to learn feasible parameters for *any model* parameterized by N Kraus operators with $\kappa \in \mathbb{C}^{nN \times n}$.

Algorithm 2 Learning HQMMs using Constrained Optimization on the Stiefel Manifold

Input: Training data $Y \in \mathbb{N}^{M \times T}$, where M is the # of data points and T is the length of the observed sequences

Hyperparameters: τ (learning rate), α: (learning rate decay), B (number of batches), E (number of epochs)

Output: Kraus operators $\{\hat{K}_i\}_{i=1}^N$

Initialize: Complex orthonormal matrix on Stiefel manifold $\kappa \in \mathbb{C}^{nN \times n}$ and partition into Kraus operators $\{\hat{K}_i\}_{i=1}^N$, with $\hat{K}_i \in \mathbb{C}^{n \times n}$
for *epoch* $= 1:\ E$ **do**
  Partition training data Y into B batches $\{Y_b\}$
  for $b = 1{:}B$ **do**
    Compute gradient $G_i \leftarrow \frac{\partial \mathcal{L}}{\partial \hat{K}_i^*}$ for batch Y_b and loss function $\mathcal{L}$ for the QGM
    Compute $\frac{\partial \mathcal{L}}{\partial \kappa} = G \leftarrow \begin{bmatrix} G_1 & \cdots & G_N \end{bmatrix}^T$
    Construct $U \leftarrow [\ G \mid \kappa\ ]$, $V \leftarrow [\ \kappa \mid -G\ ]$
    Update $\kappa \leftarrow \kappa - \tau U \left(\mathbb{I} + \frac{\tau}{2} V^\dagger U\right)^{-1} V^\dagger \kappa$
  end for
  Update learning rate $\tau = \alpha\tau$
  Re-partition κ into $\{\hat{K}_i\}$
end for
return $\{\hat{K}_i\}$

2.5.7 Empirical Results

We now share our results from Adhikary et al. [2019], evaluating our proposed method for learning HQMMs via constrained optimization on the Stiefel manifold (**COSM**) with the Wen-Yin algorithm. We compare COSM to our initial proposal of searching for local Givens rotations that increased likelihood (**GS**), as well as with HMMs learned with the Expectation-Maximization (**EM**) algorithm. We evaluate these methods on two datasets. The first is a synthetic dataset that was generated by an HMM. The second is a real-world dataset, on which the GS approach is prohibitively slow; demonstrating the scalability of COSM.

Training We train our HQMMs by optimizing negative log-likelihood loss function from Equation 2.26. We initialize the latent state $\hat{\rho}_0$ as a random Hermitian PSD matrix using the QETLAB toolbox [Johnston, 2016], and κ as a random orthonormal matrix. Except for very small models, COSM is fairly robust to random initializations (see Appendix A.1.2). We compute the gradient of the loss function with respect to the complex conjugate of the Kraus operators using the Autograd package[11] (which can handle complex differentiation), and vertically stack the gradients of the Kraus operators to construct the gradient G of the matrix κ. To smoothen the trajectory we apply momentum with $\beta = 0.9$ [Rumelhart et al., 1988, Qian, 1999], and re-normalize the gradient before and after the momentum update, making the magnitude of updates entirely dependent on step-size. We refer to HQMM size using the tuple $(n, s, w)-$HQMM, where n is the number of hidden states, s is the number of possible outputs, and w is the dimension of the 'environment' system determining the number of Kraus operators per observation (see Section 2.5.2). Consequently, for an $(n, s, w)-$HQMM we have the stacked matrix of Kraus operators $\kappa \in \mathbb{C}^{nsw \times n}$. We also provide the performance of HMMs trained using the Expectation-Maximization (EM) algorithm (with 5 random restarts) for comparison. Details of our hyperparameter tuning procedure and computing infrastructure are described in Appendix A.1.3.

Metrics On the synthetic HMM dataset, we use a scaled log-likelihood [Zhao and Jaeger, 2007] independent of sequence length called description accuracy: $DA = f\left(1 + \frac{\log_s P_\theta(Y_1,\ldots,Y_\ell)}{\ell}\right)$, where $f(\cdot)$ squishes the log-likelihood from $(-\infty, 1]$ to $(-1, 1)$. When $DA = 1$, the model predicted the sequence with perfect accuracy, and when $DA > 0$, the model performed better than random. The error bars represent one standard deviation of the DA scores across many test samples. On the real-world dataset, we report the average accuracy for a classification problem.

[11] `https://github.com/HIPS/autograd`

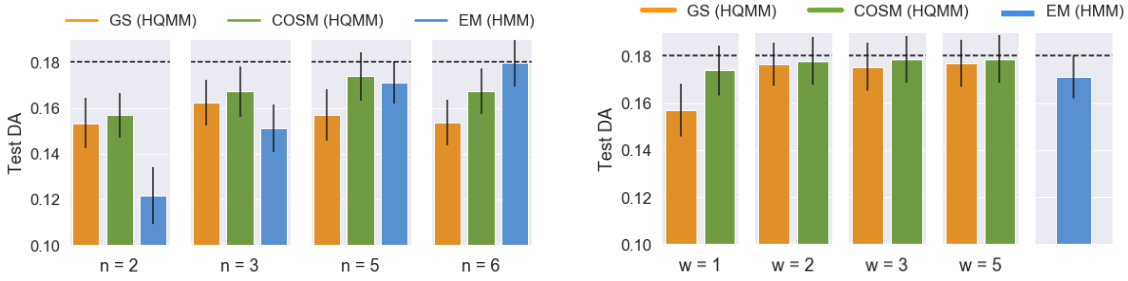
Synthetic HMM Data

For our first experiment, we generated data using a synthetic HMM with 6 hidden states and 6 possible outputs (a '(6,6)-HMM'):

$$\mathbf{A} = \begin{bmatrix} 0.8 & 0.01 & 0 & 0.1 & 0.3 & 0 \\ 0.02 & 0.02 & 0.1 & 0.15 & 0.05 & 0 \\ 0.08 & 0.03 & 0.1 & 0.4 & 0.05 & 0.5 \\ 0.05 & 0.04 & 0.5 & 0.35 & 0 & 0.5 \\ 0.03 & 0.5 & 0.03 & 0 & 0.6 & 0 \\ 0.02 & 0.4 & 0.27 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 0.2 & 0 & 0.05 & 0.95 & 0.01 & 0.05 \\ 0.7 & 0.1 & 0.05 & 0.01 & 0.05 & 0.05 \\ 0.05 & 0.8 & 0.1 & 0.02 & 0.05 & 0.04 \\ 0.04 & 0.04 & 0.02 & 0 & 0.84 & 0.11 \\ 0.01 & 0.03 & 0.7 & 0.01 & 0.02 & 0.2 \\ 0 & 0.03 & 0.08 & 0.01 & 0.03 & 0.55 \end{bmatrix} \tag{2.31}$$

We show two things with the experiments on this dataset: 1) COSM achieves better performance than GS, and 2) COSM is much faster than GS – so much so that we could train larger HQMMs than with GS. Finally, we investigate the effects on increasing model size by adding latent states (n) versus increasing the dimension of the environment variable (w).

We used 20 training and 10 validation sequences of length 3000, splitting up each sequence into 300 sequences and using a burn-in of 100. We trained HQMMs using the COSM approach for 60 epochs, and evaluated the model with the highest validation DA score on the test set. The results for this model are shown in Figure 2.9a and Figure 2.9b.



(a) $w = 1$ (b) $n = 5$

Figure 2.9: Test Set Performances on the Synthetic HMM Data: The dashed line represents the test set performance of the true model (a (6,6)-HMM) that generated the data.

COSM finds better optima than GS As shown in Figure 2.9a, HQMMs (with $w = 1$) learned using COSM achieve better performance than HQMMs learned using GS for all n. As described in Section 2.5.2, an HQMM with $w = 1$ is essentially a complex-valued NOOM. We also find that small HQMMs ($n \leq 5$) can model this data better than small HMMs, but HMMs beat HQMMs when using the true number of hidden states $n = 6$. However, we can take advantage of the additional mode w available to HQMMs to further improve performance, as shown in Figure 2.9b for $(5, 6, w)-$HQMMs (varying w). Also note that the number of parameters for an HQMM scales faster than for an HMM.

COSM is much faster than GS In Figure 2.10, we plot the test set DA versus CPU training time for the smallest and largest models trained. To ensure a fair comparison, we train both approaches on sequences of length 300 and a batch size of 30. Note that we pre-tune hyperparameters on the validation set, and the graphs show the changing test DA as the models are trained with these hyperparameters (test DAs were not used to tune hyperparameters).

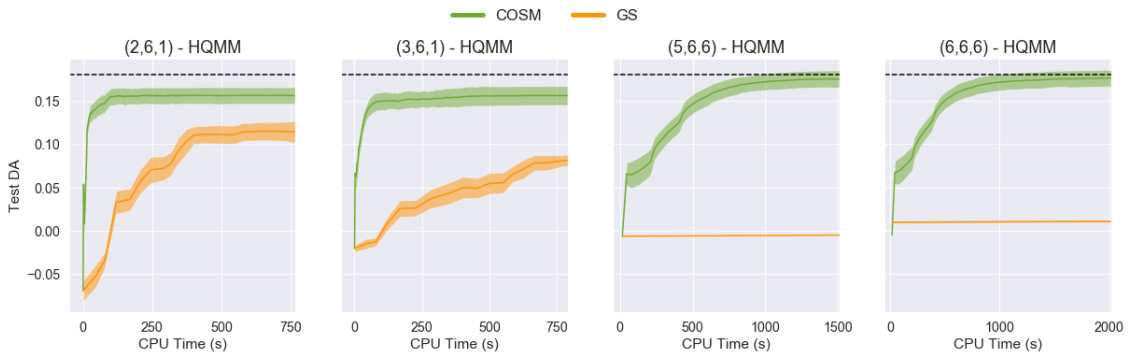


Figure 2.10: COSM Learns More Accurate Models Faster than GS: Test DA versus training time for various (n, s, w)-HQMMs trained on the synthetic HMM data. COSM converges to a better result faster than GS for all models; the dashed line represents the DA of the true data generating model.

For all models, we see that COSM converges much faster than GS, and the difference in both speed and accuracy is especially pronounced for the larger models; COSM converges within a few hundred seconds, while GS yields very poor solutions even after 2000 seconds. As the GS method can take days to converge for large models, we could not directly

calculate a precise speedup. Our derivation in Section 2.5.2 implies that a $(6, 6, 6)$-HQMM should be sufficient to fully model a $(6, 6)$-HMM, but the GS method was too slow to train this model. With COSM, we are able to show that this theoretical guarantee holds in practice. In fact, we find that in practice a $(5, 6, 3)-$HQMM is able to match the performance of the $(6, 6)-$HMM (Figure 2.9b)!

Splice Dataset

For our second experiment, we use the real-world splice dataset [Dheeru and Karra Taniskidou, 2017, Towell et al., 1991] consisting of DNA sequences of length 60, each element of which represents a nucleobase. These DNA nucleobases fall into one of four categories: Adenine (A), Cytosine (C), Guanine (G), and Thyamine (T). A DNA sequence typically consists of information encoded in sub-sequences (exons), that are separated by superfluous sub-sequences (introns). The task associated with this dataset is to classify sequences as having an exon-intron (EI) splice, an intron-exon (IE) splice, or neither (N), with 762, 765, and 1648 labeled examples for each label respectively. In addition to A, C, T and G, the raw dataset also contains some ambiguous characters, which we filter out prior to training. Our goal in this experiment is to demonstrate that we can use COSM to train HQMMs on real-world datasets (for which GS would be too slow).

We train a separate model for each of the three labels, and during test-time, choose the label corresponding to the model that assigned the highest likelihood to the given sequence. We train HQMMs using the COSM method and HMMs with the EM algorithm (with 5 random restarts) for reference. In Figure 2.11, we report the average classification accuracies across all labels obtained with 5-fold cross validation. For reference, a random classifier achieves around 33.3% accuracy.

Computing 5-fold cross-validation is prohibitively time consuming for GS, even for models with a modest number of parameters. However, we are able to learn these HQMMs with COSM. We also see that, (as before) there is a sizable marginal gain in DA when going from $w = 1$ to $w = 2$, with the benefits of increasing w further being less clear.

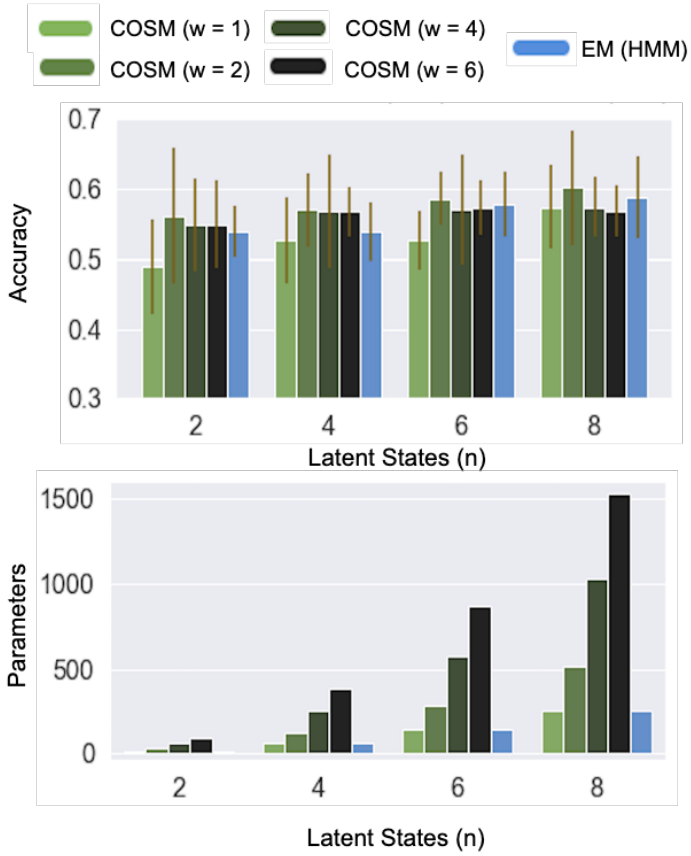


Figure 2.11: Average 5-fold Test Set Performance on the Splice Dataset: Test set accuracies (top) and number of parameters (bottom) for various HQMMs and HMMs trained using the COSM and EM algorithms respectively. Error bars in the left graph represent the mean standard deviation across labels over the 5 folds.

However unlike the previous experiment, we still see persistent gains by increasing n. Interpreting this in conjunction with the results in the previous section suggests that we have to tune both n and w depending on the dataset. We also find that for a given number of hidden states, COSM is able to learn an HQMM that outperforms the corresponding HMM, although this comes at the cost of a rapid scaling in the number of parameters.

Main Takeaways from Empirical Results

- The COSM method is a tractable approach to estimating the parameters of an HQMM from data, and outperforms the GS method (adapted from the NOOM learning algorithm of Zhao and Jaeger [2007]) in both speed and accuracy.

- Small HQMMs (with few hidden states) can outperform small HMMs learned with EM. However, when HMMs are sufficiently expressive to capture the process, HMMs perform just as well as HQMMs.

- The parameter count for HQMMs scales much faster when increasing the number of latent states (since the state representation is a matrix). The edge that small HQMMs have over small HMMs with the same number of 'latent states' no longer holds when considering the true parameter count.

- Despite the greater *theoretical* expressiveness of HQMMs, we do not have an example of real-world classical problem on which HQMMs would clearly be the preferred model. It may well be the case that all such processes are fundamentally quantum mechanical, so HQMMs are only of use in modeling explicitly quantum systems and add no value to modeling to any classical stochastic process.

2.6 Kernel HQMMs

Thus far, we have looked at *discrete*-state HQMMs, motivated by a search for a more expressive and learnable finite-dimensional extension of discrete HMMs. While HQMMs are formulated for discrete observations in finite-dimensional state spaces, the formalism of quantum channels and density operators holds for continuous, infinite-dimensional state spaces as well. Additionally, since quantum states already live in a Hilbert space, this gives a natural connection to work on embeddings [Schuld and Killoran, 2019] of probability distribution in an RKHS [Smola et al., 2007] where the idea is to treat probability distributions as points in Hilbert space. In our work Srinivasan et al. [2018a], we develop general 'quantum' analogues of operations like conditioning and marginalizing, extending HQMMs to both discrete and continuous observations using kernel methods. An interesting result is that a quantum-inspired construction of kernel HQMMs suggests a Bayesian update rule that is *different* from the standard kernel Bayes' rule [Song et al., 2009]; it is computationally cheaper and resembles Nadaraya-Watson kernel regression. We give

some preliminaries on embedding probability distributions in reproducing kernel Hilbert space, and discuss how our approach allows us to manipulate probability distributions in Hilbert space using HQMM update rules.

2.6.1 Background

RKHS Preliminaries Suppose we have data in some space $\mathcal{X}$. A kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is said to be positive definite if it satisfies $\sum_{i,j}^{n} c_i c_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0$ for any collection $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$ and any $c_1, \ldots c_n \in \mathbb{R}$. The Moore-Aronszajn theorem guarantees that every symmetric positive definite kernel $k(\mathbf{x}_i, \mathbf{x}_j)$ defines a unique feature Hilbert space $\mathcal{H}$ such that the kernel function evaluation is equivalent to taking inner products of features of the arguments in this Hilbert space, i.e., $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle_{\mathcal{H}}$ for any $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$ where $\phi : \mathcal{X} \to \mathcal{H}$ is the canonical feature map of the kernel. This Hilbert space is a *reproducing kernel Hilbert space* because it is a complete inner product space defined by the kernel and has the reproducing property: for any $f : \mathcal{X} \to \mathbb{R}$ where $f \in \mathcal{H}$ the function evaluation can be written as an inner product, i.e., $f(x) = \langle f, \phi(x) \rangle_{\mathcal{H}}$.

Mean Embeddings and Cross-Covariance Operators Smola et al. [2007] and Song et al. [2009] go beyond merely mapping individual data points to an RKHS and develop the theory of embedding *probability distributions* in an RKHS using the mean map: for a given probability measure $\mathbb{P}$ over $\mathcal{X}$ the kernel mean embedding of the probability distribution is $\mu_{\mathbb{P}} = \int k(\mathbf{x}, \cdot) \, d\mathbb{P}(\mathbf{x}) = \mathbb{E}_X[\phi(\mathbf{x})]$. Empirically, we only have access to a finite sample $\mathbf{x}_1, \ldots, \mathbf{x}_n$ so the empirical mean embedding is estimated as $\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} \phi(\mathbf{x}_i)$. We can also estimate the embeddings of joint distributions using *cross-covariance operators* defined as $C_{YX} := \mathbb{E}_{YX}[\phi(Y) \otimes \phi(X)]$, with empirical cross covariance operators estimated as $\hat{\mathcal{C}}_{YX} = \frac{1}{n} \sum_{i=1}^{n} \phi(\mathbf{y}_i) \otimes \phi(\mathbf{x}_i)$ when we have a finite sample $\{(\mathbf{y}_1, \mathbf{x}_1), (\mathbf{y}_2, \mathbf{x}_2), \ldots, (\mathbf{y}_n, \mathbf{x}_n)\}$.

Embeddings of Conditional Distributions Song et al. [2009] go beyond embedding probability distributions to develop the theory of embedding *conditional distributions*. They define a conditional embedding operator $\mathcal{C}_{Y|X} : \mathcal{H}_X \to \mathcal{H}_Y$ that satisfies two condi-

tions: 1) it must map the features of an input $\phi(x)$ to the conditional mean embedding $\mu_{Y|x}$ as $\mu_{Y|x} = \mathcal{C}_{Y|x}$, and 2) this resulting conditional mean embedding should allow evaluation of conditional expectations via inner products as $\mathbb{E}_{Y|x}[g(Y)|x] = \langle \mu_{Y|x}, g \rangle$ for all $g \in \mathcal{H}_X$. They find these conditions are satisfied when setting $\mathcal{C}_{Y|X} = \mathcal{C}_{YX}\mathcal{C}_{XX}^{-1}$ as long as the conditional expectation function is in the original RKHS, i.e., $\mathbb{E}_{Y|x}[g(Y)|\cdot] \in \mathcal{H}_X$. If this assumption is violated, the conditional embedding operator is treataed as producing an *approximation* to the conditional mean embedding $\mu_{Y|x}$.

Dynamical Systems in Hilbert Space Having established that probability distributions can be represented as points in Hilbert space, Song et al. [2010] and Boots et al. [2013] further develop this theory to *manipulate* conditional distributions in Hilbert space using HMM and PSR update rules. The idea is that HMM/PSR update rules are an application of elementary probabilistic operations (conditioning, marginalization, constructing joint probabilities). Thus, they propose a kernel sum rule, kernel product rule, and kernel Bayes' rule to perform these operations *directly on mean embeddings in Hilbert space*, to maintain and manipulate update entire probability distributions directly in Hilbert space.

Then, in light of the work on HQMMs presented in Section 2.5.2, a natural question that arises is whether there is anything to be gained by updating Hilbert space embeddings with HQMM update rules instead of HMM/PSR update rules. In the case that we use a feature map $\phi(\cdot)$ of an infinite-dimensional Hilbert space, there is no clear *model expressiveness* motivation to prefer HQMM update rules. For finite dimensional feature maps, we may be able justify HQMM update rules *if* the feature map produces a probability distribution (since the model expressiveness relationships we previously explored are only relevant for linear *stochastic* processes). Such motivation remains murky; instead the most intriguing reason is that the HQMM formulation suggests a computationally cheaper Bayesian update rule that appears analogous to Nadaraya-Watson kernel regression. The exact nature of this correspondence remains a direction for future investigation.

2.6.2 Quantum Mean Embeddings

The first step to formulating kernel HQMMs is to develop a notion of quantum mean embeddings. Kübler et al. [2019] propose such a *quantum mean embedding* (QME) of distributions, with the idea being to map probability distributions not just to arbitrary points in Hilbert space, but ones that correspond to *physical quantum systems*. Such embeddings begin with a choice of feature map $\phi(\cdot)$ that map data to a (normalized) quantum system, which also defines the kernel with the requirement that $k(\mathbf{x}, \mathbf{x}) = \langle \phi(\mathbf{x}), \phi(\mathbf{x}) \rangle_{\mathcal{H}} = 1$. With this, they define the quantum mean embedding $\nu_{\mathbb{P}} = \frac{1}{\mathcal{N}_{\mathbb{P}}} \int_{\mathcal{X}} \phi(\mathbf{x}) \; d\mathbb{P}(\mathbf{x}) = \frac{1}{\mathcal{N}_{\mathbb{P}}} \mathbb{E}_{\mathbf{x} \sim \mathbb{P}}[\phi(\mathbf{x})]$ with the normalization factor $\mathcal{N}_{\mathbb{P}} = \| \mathbb{E}_{\mathbf{x} \sim \mathbb{P}}[\phi(\mathbf{x})] \|_2$ to ensure that the embedding itself is a valid quantum state. Essentially, they propose a feature map that maps data on to a high-dimensional unit sphere. They show that for universal kernels (those whose RKHS is dense in the space of bounded continuous functions on $\mathcal{X}$), the quantum mean embedding is injective and captures the full information about the distribution being embedded. Their discussion focuses on encoding data into quantum states so that they may be analysed on physical quantum systems [Schuld and Killoran, 2019].

The feature map of Kübler et al. [2019] maps data to the unit sphere, which is analogous to representing data with *pure states*. Yet, as we discussed in Section 2.5.1, mixed states (represented by density matrices) are a more general notion of quantum state. Accordingly, our proposal maps data to density operators in the tensor product space $\mathcal{H} \otimes \mathcal{H}$ which is itself a Hilbert space equipped with the Hilbert-Schmidt inner product.

Definition 10 (Quantum Mixture Mean Embedding)**.** *Let $\phi : \mathcal{X} \to \mathcal{H}$ be the canonical feature map of a kernel satisfying $\|\phi(x)\|^2_{\mathcal{H}} = 1$. We define $\tilde{\phi}(\mathbf{x}) := \phi(\mathbf{x}) \otimes \phi(\mathbf{x})$, and the quantum mixture mean embedding as $\rho_\pi := \int_{\mathcal{X}} \phi(\mathbf{x}) \otimes \phi(\mathbf{x}) \; d\pi = \mathbb{E}_\pi[\phi(\mathbf{x}) \otimes \phi(\mathbf{x})] = \mathbb{E}_\pi[\tilde{\phi}(\mathbf{x})]$. We also have the quantum mixture cross-covariance operator $\Lambda^\pi_{XY} = \mathbb{E}_\pi[\tilde{\phi}(\mathbf{x}) \otimes \tilde{\phi}(\mathbf{y})]$.*

We can also understand the quantum mixture mean embedding (QMME) ρ_π as the covariance operator $\mathcal{C}^\pi_{XX}$ with quantum feature map $\phi(\cdot)$, and the quantum mixture covariance operator (QMCO) Λ^π_{XY} as a density operator in a tensor product space. Note that

we altogether avoid the need for the normalization factor $\mathcal{N}_{\mathbb{P}}$ from Kübler et al. [2019]. This is because the quantum mixture mean embedding maps data to density matrices in a tensor product Hilbert space, i.e., to $\rho_{\mathbf{x}} = \tilde{\phi}(\mathbf{x}) = \phi(\mathbf{x}) \otimes \phi(\mathbf{x})$ where $\|\rho_{\mathbf{x}}\|_{\text{HS}} = \text{tr}(\rho_{\mathbf{x}}) = 1$, as opposed to pure quantum states $\phi(\mathbf{x})$ with $\|\phi(\mathbf{x})\|_2 = 1$, per the quantum mean embedding suggested by Kübler et al. [2019]. This QME requires re-normalization of the mean embedding to ensure it is a valid quantum state (i.e., $\nu_{\mathbb{P}} = \frac{1}{\|\mu_{\mathbb{P}}\|_{\mathcal{H}}} \mu_{\mathbb{P}} = \frac{1}{\|\mu_{\mathbb{P}}\|_{\mathcal{H}}} \mathbb{E}_{\mathbf{x} \sim \mathbb{P}}[\phi(\mathbf{x})]$ so that $\|\nu_{\mathbb{P}}\|_2 = 1$), there is no need to re-normalize our quantum mixture mean embedding since $\rho_{\mathbb{P}} = \mathbb{E}_{\mathbf{x} \sim \mathbb{P}}[\phi(\mathbf{x}) \otimes \phi(\mathbf{x})]$ already satisfies $\|\rho_{\mathbb{P}}\|_{\text{HS}} = 1$ (averaging over density operators gives a density operator). It also trivially follows that if the feature map $\phi(\cdot)$ corresponds to a universal kernel (and so a characteristic kernel by Theorem 1 of Kübler et al. [2019]), the QMME with feature map $\tilde{\phi}$ is also injective.

We remark that if we use the QME as the embedding of a belief state and develop a scheme to manipulate this embedding in Hilbert space, we would formulate NOOM updates in Hilbert space. On the other hand, if we use the QMME to embed belief states in Hilbert space, the natural rules for manipulating this embedding the tensor product Hilbert space produce HQMM updates. We refer to the use of the QMME to embed belief states and the use of HQMM equations to manipulate the embedding in Hilbert space as *kernel HQMMs*, and develop this model below.

2.6.3 Kernel HQMMs

We can use the QMME to map any belief state to a quantum system (as a density operator) in Hilbert space, but two natural questions arise: 1) how can we represent transition dynamics to *update the embedding* of the belief state, and 2) how can we *condition the embedding* on observation? The solution is to develop an analog of the 'sum rule' and Bayes' rule operations from probabilistic graphical models. Srinivasan et al. [2018a] present a lengthy derivation, starting from quantum circuits and showing that the HQMM equations amount to using the kernel sum rule for state transitions and a Nadaraya-Watson-like approach for conditioning on observations. We use our L-HQMM formulation from Defi-

nition 9 to provide a simpler and more straightforward formulation of kernel HQMMs.

Sum Rule

In the classical case, given a probability distribution $P(X)$ represented by a vector $\vec{x}$ and a likelihood $P(Y|X)$ represented by the column stochastic matrix $\mathbf{A}$, the *sum rule* specifies that we can compute $\vec{y} := P(Y) = \sum_x P(Y|x)P(x) = \mathbf{A}\vec{x}$. In an RKHS, the *kernel sum rule* [Song et al., 2009] relates mean embeddings μ_X^π, μ_Y^π of distributions $P(X)$ and $P(Y)$[12] respectively using the conditional embedding operator $\mathcal{C}_{Y|X}$ as $\mu_Y^\pi = \mathcal{C}_{Y|X}\mu_X^\pi$.

Building on the discussion in Section 2.5.1, the analogous 'quantum' sum rule is this: given a density operator of a QMME in Hilbert space $\vec{\rho}_X \in \mathcal{H}_X$ and a Liouville superoperator $\mathbf{L}_{Y|X}$ (representing a quantum channel $\mathcal{E} : \mathcal{H}_X \to \mathcal{H}_Y$), we can compute $\vec{\rho}_Y \in \mathcal{H}_Y$ as:

$$\vec{\rho}_Y = \mathbf{L}_{Y|X}\vec{\rho}_X \tag{2.32}$$

This is exactly identical to the kernel sum rule, but there is a subtle feature of the quantum case. The conditional embedding operator was defined $\mathcal{C}_{Y|X} := \mathcal{C}_{YX}\mathcal{C}_{XX}^{-1}$ so it satisfied certain desirable properties (see Section 2.6.1), and Grünewälder et al. [2012] show that this can be interpreted as the solution to a regression problem. In our case, the Liouville superoperator must satisfy those desirable properties, and it should *also* satisfy the CP-TP conditions in Definition 9 as it is a map between quantum systems. In practice, we define the Liouville superoperator the same way as the conditional embedding operator: as $\mathbf{L}_{Y|X} = \Lambda_{YX}\Lambda_{XX}^{-1}$ where Λ_{YX} is the quantum mixture cross-covariance operator. The superoperator estimated this way via regression will still be TP (since it is a linear constraint), but owing to the operator inversion we are not guaranteed a CP superoperator in general[13].

Now, we can specify how to update the QMME of a belief state to account for transition

[12] These need not be the marginals of the joint distribution, but can be any prior distribution.

[13] In the previous section, we estimated CPTP superoperators through constrained optimization of the likelihood and ensuring that the Kraus operators lay on the Stiefel manifold. Here, we are working with *embeddings* of distributions in an RKHS, which are just a representation of the underlying distribution. We do not have access to the underlying pdf, so we cannot compute likelihoods and use a similar constrained optimization approach to guarantee learning CP maps.

dynamics: for quantum mixture mean embeddings $\vec{\rho}_t, \vec{\rho}_{t-1}$ of belief states, the 'quantum' sum rule dictates that the continuous belief state (represented as a vectorized density matrix) may be updated via Liouville operator $\mathbf{L}_{X_t|X_{t-1}} = \Lambda_{X_t X_{t-1}} \Lambda^{-1}_{X_{t-1} X_{t-1}}$ as $\vec{\rho}\,'_t = \mathbf{L}_{X_t|X_{t-1}} \vec{\rho}_{t-1}$.

Bayes' rule

In the classical case, given a prior probability distribution $Q(X)$ represented by a vector $\vec{x}$, a conditional probability table $P(Y|X)$ represented by matrix $\mathbf{A}$, and an observation y, we perform Bayesian update as follows:

$$P(X|y) = \frac{P(y|X)P(X)}{\sum_x (y|x)P(x)} = \frac{\text{diag}(\mathbf{A}_{(y,:)})\vec{\pi}}{\mathbb{1}^T \text{diag}(\mathbf{A}_{(y,:)})\vec{\pi}} \tag{2.33}$$

In an RKHS, given mean embedding μ^π_X and covariance $\mathcal{C}^\pi_{XX}$ of a prior belief state and features of an observation $\phi(y)$, kernel Bayes' rule [Song et al., 2013] specifies how to find the updated mean embedding $\mu^\pi_{X|y}$ conditioned on observation: as $\mu^\pi_{X|y} = (\mathcal{C}^\pi_{XY} C^\pi_{YY})^{-1} \phi(y)$, where $C^\pi_{YY} = \mathcal{C}_{(YY)|X} \mu^\pi_X$ and $\mathcal{C}^\pi_{XY} = (\mathcal{C}_{Y|X} \mathcal{C}^\pi_{XX})^{-1}$. Now, given a QMME $\vec{\rho}_X$ of a prior belief state and features of an observation $\tilde{\phi}(y)$, we could construct an exact quantum-analogue of the kernel Bayes' rule just as we did for the sum rule. Yet, the more natural way to set up Bayes' rule in an RKHS when the systems are quantum (given quantum mixed covariance operator Λ_{XX}) is as:

$$\rho_{X|y} = \frac{\left(\mathbf{L}_{Y|X} \Lambda_{XX}\right)^\dagger \tilde{\phi}(y)}{\| \left(\mathbf{L}_{Y|X} \Lambda_{XX}\right)^\dagger \tilde{\phi}(y) \|_1} \tag{2.34}$$

where $\mathbf{L}_{Y|X} = \Lambda_{YX} \Lambda^{-1}_{XX}$ as before and $\| \cdot \|_1$ is the trace norm (which is finite since QMME and QMCO are trace-class by construction). The reason this is natural is because it exactly mirrors the quantum circuit analogue of Bayesian updating in Figure 2.5a. First, Λ_{XX} is an encoding of the prior distribution $Q(X)$ as a density operator. Second, $\mathbf{L}_{Y|X}$ maps this density operator prior to a joint density operator encoding the joint distribution $Q(X, Y)$ (essentially the *kernel chain rule*). Finally $\tilde{\phi}(y)$ projects the joint density operator

onto the subspace corresponding to the observation, and the denominator 'renormalizes' the joint density operator. Applying this to our dynamical system model, the update rule to condition Λ_{tt} on observation y_t is $\vec{\rho}_{t|y_t} = \frac{\left(\mathbf{L}_{Y_t|X_t}\Lambda_{tt}\right)^\dagger \tilde{\phi}(y_t)}{\|\left(\mathbf{L}_{Y_t|X_t}\Lambda_{tt}\right)^\dagger \tilde{\phi}(y_t)\|_1}$.

Finite Sample Kernel Estimates

Suppose we collect data from the joint distribution $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ as well as from the prior $\{\tilde{x}_1, \ldots, \tilde{x}_n\}$. Let $\Upsilon_X = \begin{pmatrix} \tilde{\phi}(x_1) & \ldots & \tilde{\phi}(x_n) \end{pmatrix}$, $\tilde{\Upsilon}_X = \begin{pmatrix} \tilde{\phi}(\tilde{x}_1) & \ldots & \tilde{\phi}(\tilde{x}_n) \end{pmatrix}$, $\Phi_Y = \begin{pmatrix} \tilde{\phi}(y_1) & \ldots & \tilde{\phi}(y_n) \end{pmatrix}$ with $\Upsilon_X, \tilde{\Upsilon}_X, \Phi_Y \in \mathbb{C}^{d\times n}$ where d is the dimension of the RKHS (potentially infinite). We note that by the representer theorem [Schölkopf et al., 2001, Muandet et al., 2016], we can write the empirical estimate of the QMME as $\tilde{\rho}_X = \tilde{\Upsilon}_X\tilde{\alpha}_X$, $\rho_X = \Upsilon_X\alpha_X$, and $\rho_Y = \Phi_Y\alpha_Y$ for some weights $\tilde{\alpha}_X, \alpha_X, \alpha_Y \in \mathbb{R}^n$ on the data points. Additionally, the empirical estimate of the conditional embedding operator $\mathbf{L}_{Y|X} = \Lambda_{YX}\Lambda_{XX}^{-1}$ can be written as $\Phi_Y(\Upsilon_X^\dagger\Upsilon_X + \lambda\mathbb{I})^{-1}\Upsilon_X^\dagger$, and the empirical estimate of $\mathbf{L}_{Y|X}\Lambda_{XX}$ can be written as $\Phi_Y D\Upsilon_X^\dagger$ where $D = \text{diag}\left(\left(\Upsilon_X^\dagger\Upsilon_X + \lambda\mathbb{I}\right)^{-1}\Upsilon_X^\dagger\tilde{\Upsilon}_X^\dagger\tilde{\alpha}_X\right)$. λ is a regularizer to ensure the inverse is well-defined.

Sum Rule Given some weights $\tilde{\alpha}_X$ on $\tilde{\Upsilon}_X$, the kernelized sum rule is $(\Phi_Y\alpha_Y) = \left(\Phi_Y(\Upsilon_X^\dagger\Upsilon_X + \lambda\mathbb{I})^{-1}\Upsilon_X^\dagger\right)\left(\tilde{\Upsilon}_X\tilde{\alpha}_X\right)$. Using Gram matrices $(K_{XX})_{ij} = k(x_i, x_j) = \langle\tilde{\phi}(x_i), \tilde{\phi}(x_j)\rangle$ and $\left(\tilde{K}_{XX}\right)_{ij} = k(x_i, \tilde{x}_j) = \langle\tilde{\phi}(x_i), \tilde{\phi}(\tilde{x}_j)\rangle$, we can write the weights α_Y over Φ_Y from performing the kernel 'quantum' sum rule as:

$$\alpha_Y = (K_{XX} + \lambda\mathbb{I})^{-1}\tilde{K}_{XX}\tilde{\alpha}_X \tag{2.35}$$

Bayes' rule The kernelized Bayes' rule returns the empirical conditional QMME $\hat{\rho}_{X|y} = \Upsilon_X\beta_X$ for some weights $\beta_X \in \mathbb{R}^n$ upon observation y, and must obey:

$$\Upsilon_X\beta_X = \frac{\left(\Phi_Y D\Upsilon_X^\dagger\right)^\dagger\tilde{\phi}(y)}{\|\left(\Phi_Y D\Upsilon_X^\dagger\right)^\dagger\tilde{\phi}(y)\|_1} = \frac{\Upsilon_X D\Phi_Y^\dagger\tilde{\phi}(y)}{\|\Upsilon_X D\Phi_Y^\dagger\tilde{\phi}(y)\|_1} = \Upsilon_X\frac{Dk_y}{\mathbb{1}^T Dk_y} \tag{2.36}$$

where $(\tilde{k}_y)_j = k(y_j, y) = \langle\tilde{\phi}(y_j), \tilde{\phi}(y)\rangle$. The denominator simplifies because $\|\Upsilon_X Dk_y\|_1 =$

$\| \sum_{i=1}^n \tilde{\phi}(x_i)(Dk_y)_i \|_1 = \text{tr}\left(\sum_{i=1}^n \tilde{\phi}(x_i)(Dk_y)_i \right) = \sum_{i=1}^n \text{tr}\left(\tilde{\phi}(x_i) \right)(Dk_y)_i = \sum_{i=1}^n (Dk_y)_i = \mathbb{1}^T Dk_y$. Thus, we can write the weights β_X over Υ_Y from performing the kernel 'quantum' Bayes' rule as:

$$\beta_X = \frac{Dk_y}{\mathbb{1}^T Dk_y} \tag{2.37}$$

Connection to Nadaraya-Watson Kernel Regression We now discuss the idea that this 'quantum' Bayesian update can be viewed as a quasi prior-weighted Nadaraya-Watson kernel regression. Nadaraya-Watson kernel regression is a non-parametric regression method that computes the conditional mean of x given y as $\hat{m}(y) = \vec{x}^T \frac{k_y}{\mathbb{1}^T k_y}$ where each element of vector $(k_y)_i = k(y_i, y)$ and $\vec{x} \in \mathbb{R}^n$ is a vector of data points. The idea is to predict x from y by weighting every x_i in the dataset in proportion to how similar its corresponding y_i is to y, as given by the kernel. Clearly, using the weights $\beta_X = \frac{Dk_y}{\mathbb{1}^T Dk_y}$ from Equation 2.37 to compute conditional mean embedding $\rho_{X|y} = \Upsilon_X \frac{Dk_y}{\mathbb{1}^T Dk_y}$ is nearly identical to doing Nadaraya-Watson kernel regression from observation to the conditional embedding of the posterior: the only apparent difference is that the features of each data point $\tilde{\phi}(x_i)$ in Υ_X are weighted not only by how similar the corresponding y_i is to y (i.e., $k(y_i, y)$), but also based on how similar that data point x_i is to high-α_X-weight data points $\{\tilde{x}_i\}$ of the prior, as specified by element D_{ii} in diagonal matrix D.

However, there is an important caveat: 'kernel' here is used in a different sense; it refers to *positive* kernels, while the 'kernel' of an RKHS is *positive-definite*. In our case, observe that in addition to the kernel $k(x_i, x_j) = \langle \tilde{\phi}(x_i), \tilde{\phi}(x_j) \rangle$ being positive definite (by the Moore–Aronszajn theorem), it is also *positive*: $\langle \tilde{\phi}(x_i), \tilde{\phi}(x_j) \rangle = \text{tr}\left((\phi(x_i)\phi(x_i)^\dagger)(\phi(x_j)\phi(x_j)^\dagger) \right) = |\phi(x_i)^\dagger \phi(x_j)|^2 \geq 0$. However, the diagonal entries of diagonal matrix D (which encode weights α_Y (by Equation 2.35)) need *not* be non-negative; the matrix inverse in Equation 2.35 may result in negative weights in α_Y. This is a consequence of the fact that we had to define $\mathbf{L}_{Y|X}$ as the solution to a regression problem, and not enforce CP constraints. For this reason, our 'quantum' Bayes' rule is only a quasi Nadaraya-Watson regression. There remain a number of unanswered questions about Bayesian updates in this manner in the RKHS, and we discuss these directions for future work at the end of this subsection.

Kernel HQMM Update Equation

Putting our 'quantum' sum rule and Bayes' rule together, we get the equation to update QMMEs with HQMM update rules:

$$\vec{\rho}_t = \frac{\left(\mathbf{L}_{X_t,Y_t|X_{t-1}} \times_3 \vec{\rho}_{t-1}\right) \tilde{\phi}(y_t)}{\| \left(\mathbf{L}_{X_t,Y_t|X_{t-1}} \times_3 \vec{\rho}_{t-1}\right) \tilde{\phi}(y_t)\|_1} \tag{2.38}$$

where $\mathbf{L}_{X_t,Y_t|X_{t-1}} = \mathbf{L}_{X_tY_t|X_t}\mathbf{L}_{X_t|X_{t-1}} = \left(\Lambda_{X_tY_tX_t}\Lambda^{-1}_{X_tX_t}\right)\left(\Lambda_{X_tX_{t-1}}\Lambda^{-1}_{X_{t-1}X_{t-1}}\right)$ is a 3-mode tensor and $A \times_c b$ corresponds to tensor contraction of tensor A with vector b along mode c. In our case this means performing tensor contraction of $\mathbf{L}_{X_t,Y_t|X_{t-1}}$ with the embedding of belief state $\vec{\rho}_{t-1}$ to produce the density operator corresponding to the joint state Λ_{X_t,Y_t}. This is essentially the RKHS version of the L-HQMM update in Equation 2.25; $\left(\mathbf{L}_{X_t,Y_t|X_{t-1}} \times_3 \vec{\rho}_{t-1}\right)$ applies the transition dynamics on the belief state and prepares the embedding of the joint state $\Lambda_{X_tY_t}$, applying $\tilde{\phi}(y_t)$ projects the joint distribution to the subspace consistent with the observation, and the denominator renormalizes the resulting QMME. An alternate view is to treat $\mathbf{L}_y = \mathbf{L}_{X_t,Y_t|X_{t-1}} \times_2 \tilde{\phi}(y)$, except we don't have a fixed set of operators $\mathbf{L}_y$ but rather we compute one for each (continous) observation non-parametrically. Finally, we give the kernelized version of the HQMM update rule (where $K_{t,t}$ and $\tilde{K}_{t,t}$ are Gram matrices):

$$\alpha'_t = (K_{t-1,t-1} + \lambda\mathbb{I})^{-1}\tilde{K}_{t-1,t-1}\tilde{\alpha}_{t-1} \quad \text{and} \quad \alpha_t = \frac{\text{diag}(\alpha'_t)k_y}{\mathbb{1}^T\text{diag}(\alpha'_t)k_y} \tag{2.39}$$

Practical Implementation To use this model in practice, the first question we need to answer is what feature map to use. We note that random Fourier features [Rahimi and Recht, 2008] with sine and cosine features yield a natural feature map $\phi(\cdot)$ such that $\|\phi(x)\|_2^2 = 1$. We can then use the outer product of random Fourier features to define the feature map $\tilde{\phi}(\cdot)$, mapping data to finite-dimensional density matrices. However, note that the scaling in the dimension of the feature space of $\tilde{\phi}(\cdot)$ is unfavourable: it is quadratic in the number of RFFs. The second question is how to make predictions using the embedding of the belief state. For discrete-state HQMMs, we could simply

treat the denominator of the state update equation as the probability of an observation y (Equation 2.25). This does not exactly carry over to the kernel HQMM case (as we don't have a fixed collection of superoperators that together satisfy CPTP constraints). Nevertheless, by analogy with the discrete case we can heuristically treat the denominator term of the HQMM update equation (Equation 2.38) as a quasi-unnormalized density of an observation $f(y_t) \propto \| \left(\mathbf{L}_{X_t,Y_t|X_{t-1}} \times_3 \vec{\rho}_{t-1} \right) \tilde{\phi}(y_t) \|_1$. Thus, to make a prediction, we sample from the convex hull of our training set, compute densities as described, and take the expectation to make a point prediction as $\frac{1}{n} \sum_{i=1}^{n} y_i f_Y(y_i)$. The final question is how we to learn model parameters $\mathbf{L}_{X_t,Y_t|X_{t-1}}$. As discussed previously, $\mathbf{L}_{X_t,Y_t|X_{t-1}} = \mathbf{L}_{X_tY_t|X_t} \mathbf{L}_{X_t|X_{t-1}}$ and these superoperators can be treated as the solution to a regression problem: $\mathbf{L}_{X_tY_t|X_t} = \left(\Lambda_{X_tY_tX_t} \Lambda^{-1}_{X_tX_t} \right)$ and $\mathbf{L}_{X_t|X_{t-1}} = \left(\Lambda_{X_tX_{t-1}} \Lambda^{-1}_{X_{t-1}X_{t-1}} \right)$. In Srinivasan et al. [2018a], we follow the 2-stage regression approach of Hefny et al. [2015].

In addition to keeping track of an embedding of the belief state that can be used for downstream tasks, we can also use the embedding as a non-parametric kernel density estimator if we use a Gaussian RBF kernel [Kanagawa and Fukumizu, 2014] and draw samples using kernel herding [Chen et al., 2012]. Thus, HQMMs have a natural connection with Hilbert space embeddings of distributions, where we map our data to quantum systems in an RKHS and model sequences as emissions from an evolving latent quantum system.

Directions for Future Work

There remain a number of unanswered questions about kernel HQMMs. Firstly, is there a way to define the conditional embedding operator $\mathbf{L}_{Y|X}$ that guarantees the CP requirement? Could we define this operator as the solution to a *constrained* least-squares problem, with the constraint that the Kraus operators of the superoperator $\mathbf{L}_{Y|X}$ lie on the Stiefel manifold? If so, does the 'quantum' Bayes' rule become exactly equivalent to prior-weighted Nadaraya-Watson regression? Finally, investigating the theoretical properties of the 'quantum' Bayes' rule will be important. In particular, we would like to understand whether it approximates the same (and *correct*) embedding of the posterior

as kernel Bayes' rule, or if this is only the case under certain conditions. Answering these questions will shed further light on the viability of the 'quantum' Bayes' rule as a practical alternative to the standard kernel Bayes' rule.

2.6.4 Connections to PSRNNs

Predictive State Recurrent Neural Networks (PSRNNs) were developed by Downey et al. [2017] by embedding a Predictive State Representation (PSR) into an RKHS. The PSRNN update equation is:

$$\vec{\omega}_t = \frac{W \times_3 \vec{\omega}_{t-1} \times_2 \phi(y_t)}{||W \times_3 \vec{\omega}_{t-1} \times_2 \phi(y_t)||_F} \tag{2.40}$$

where W is a three mode tensor corresponding to the cross-covariance between observations and the state at time t conditioned on the state at time $t-1$, and ω is a factorization of a p.s.d state matrix $\mu_t = \omega\omega^T$ (so renormalizing ω by Frobenius norm is equivalent to renormalizing μ by its trace). There is a clear connection between PSRNNs and the kernel HQMMs; this matrix μ_t is what we vectorize to use as our state $\vec{\rho}_t$ in kernel HQMMs, and both kernel HQMMs and PSRNNs are parameterized (in the primal space using RFFs) in terms of a three-mode tensor (W for PSRNNs and $\mathbf{L}_{X_t,Y_t|X_{t-1}}$ for kernel HQMMs). We also note that while PSRNNs modified kernel Bayes' rule *heuristically*, we have shown that this modification can be interpreted as a generalization of Bayes' rule for quantum systems or a kind of quasi-Nadaraya-Watson kernel regression. One key difference between these approaches is that we directly use states in Hilbert space to estimate quasi-densities of observations and make predictions. By contrast PSRNNs learn an additional ad-hoc mapping from states to observations.

2.6.5 Empirical Results

Here we present our model's performance on some experiments. We test our model on the following datasets:

- **Penn Tree Bank (PTB)** Marcus et al. [1993]. We train a character-prediction

model with a train/test split of 120780/124774 characters due to hardware limitations.

- **Swimmer** Simulated swimmer robot from OpenAI gym[14]. We collect 25 trajectories from a robot that is trained to swim forward (via the cross entropy with a linear policy) with a 20/5 train/test split. There are 5 features at each time step: the angle of the robots nose, together with the 2D angles for each of it's joints.

- **Mocap** Human Motion Capture Dataset. We collect 48 skeletal tracks from three human subjects with a 40/8 train/test split. There are 22 total features at each time step: the 3D positions of the skeletal parts (e.g., upper back, thorax, clavicle).

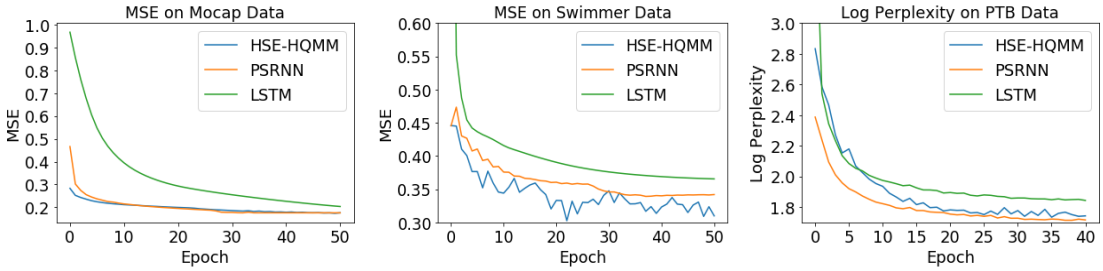


Figure 2.12: Performance of kernel HQMM, PSRNN, and LSTM on Mocap, Swimmer, and PTB

We compare the performance of three models: kernel HQMMs, PSRNNs, and LSTMs. We use RFFs as our feature map[15], and we initialize PSRNNs and kernel HQMMs using Two-Stage Regression (2SR) [Downey et al., 2017] and LSTMs using Xavier Initialization and *additionally refine all three models using Back Propagation Through Time (BPTT)*[16]. We optimize and evaluate all models on Swimmer and Mocap with respect to the Mean Squared Error (MSE) using 10 step predictions as is conventional in the robotics community. This means that to evaluate the model we perform recursive filtering on the test set to produce states, then use these states to make predictions about observations 10 steps

[14]https://gym.openai.com/

[15]However, to avoid the quadratic scaling in dimensions associated with the feature map $\tilde{\phi}(\cdot)$ of the QMME, we simply used the QME feature map $\phi(\cdot)$ and did not construct density matrices.

[16]In doing so, our model starts to stray from a kernel HQMM and work more like a PSRNN, although the way we make predictions still remains different.

in the future. We optimize all models on PTB with respect to Perplexity (Cross Entropy) using 1 step predictions, as is conventional in the NLP community. As we can see in Figure 2.12, kernel HQMMs outperform both PSRNNs and LSTMs on the swimmer dataset, and achieve comparable performance to the best alternative on Mocap and PTB.

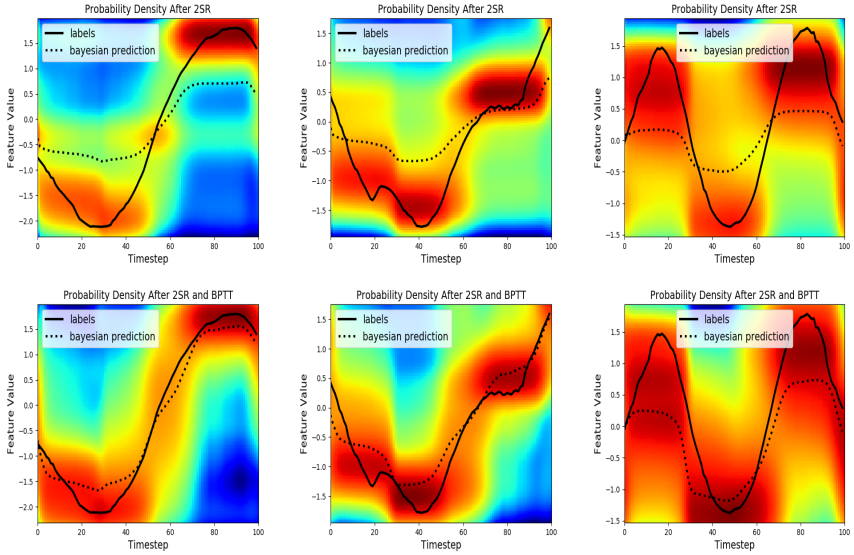


Figure 2.13: Heatmap Visualizing the quasi-densities generated by our kernel HQMM model. Red indicates high density, blue indicates low density, x-axis corresponds to time, y-axis corresponds to the feature value. Each column corresponds to the predicted marginal quasi-density of a single feature changing with time. The first row is the quasi-density after 2SR initialization, the second row is the quasi-density after the model in row 1 has been refined via BPTT.

Visualizing Quasi-Densities As mentioned previously, we can treat the denominator term in the HQMM update equation $\| \left(\mathbf{L}_{X_t, Y_t | X_{t-1}} \times_3 \vec{\rho}_{t-1} \right) \tilde{\phi}(y_t) \|_1$ as a quasi-density over observations, and we visualize this for a model trained on the Mocap dataset in Figure 2.13. We take the 22 dimensional joint density and marginalize it to produce three marginal quasi-distributions, each over a single feature. We plot the resulting marginal quasi-distributions over time using a heatmap, and superimpose the ground-truth and model predictions. We observe that BPTT (second row) improves the marginal quasi-distribution. Another interesting observation, from the the last $\sim$30 timesteps of the marginal quasi-distribution in the top-left image, is that our model is able to produce

a bi-modal distribution with probability mass at both $y_i = 1.5$ and $y_i = -0.5$, without making any parametric assumptions. This kind of information is difficult to obtain using an LSTM or PSRNN.

Parameter	Value
Kernel	Gaussian
Kernel Width	Median Neighboring Pairwise Distance
Number of Random Fourier Features	1000
Prediction Horizon	10
Batch Size	20
State Size	20
Max Gradient Norm	0.25
Number of Layers	1
Ridge Regression Regularizer	0.05
Learning Rate	0.1
Learning algorithm	Stochastic Gradient Descent (SGD)
Number of Epochs	50
BPTT Horizon	20

Table 2.2: Hyperparameter values used in experiments

2.7 Summary

After providing some background on HMMs, PSRs, and NOOMs, we presented our formulation of hidden quantum Markov models (HQMMs). We established various expressiveness relations between these model classes, and proved that HQMMs are the most expressive model class for modeling linear stochastic processes that do not also suffer from the 'negative probability problem'. We showed that HQMM parameters lie on the Stiefel manifold, and proposed an iterative maximum likelihood algorithm for learning HQMMs from data. We test this algorithm on a synthetic dataset generated by an HMM and a real-world dataset, and find that it outperforms a previous proposal of ours, but there is no clear performance advantage over HMMs learned by EM in practice. We also developed kernel HQMMs, which use HQMM-style updates to manipulate quantum mixture mean embeddings in Hilbert space.

Chapter 3

Tensor Networks for Classical and Quantum Processes

In the previous chapter, we discussed various models for linear stochastic processes such as HMMs, PSRs, NOOMs, and HQMMs. As it turns out, other approaches very similar to these models have been developed in the study of *quantum tensor networks* [Biamonte and Bergholm, 2017], a method for tractably and efficiently representing high-dimensional tensors. We formalize the links between linear stochastic processes and several quantum tensor networks, drawing on our work in Srinivasan et al. [2020].

Motivated by the exponential scaling in the representation of quantum channels in quantum process tomography, we also study the problem of developing a tensor network representation of quantum channels that is constrained to only represent physical quantum channels. This discussion draws on our work in Srinivasan et al. [2021].

3.1 Preliminaries on Tensor Networks

In our context, tensors are simply multi-dimensional arrays or multilinear maps, and a tensor network is a decomposition of (typically) a high-dimensional tensor into a representation with (typically) fewer parameters. This can be thought of as a generalization of matrix decompositions like singular-value decomposition (where pruning small singular

values can give a more compact representation of the matrix) to multidimensional arrays. Tensor networks are also come with a rich graphical notation: each vertex is a tensor, each edge is a mode (dimension) of a tensor, and a connected edge between two nodes is a summation (or contraction) along that mode of the tensor. We visually illustrate some tensors in Figure 3.1, 'indexing' into a tensor in Figure 3.2, and tensor contraction in Figure 3.3.

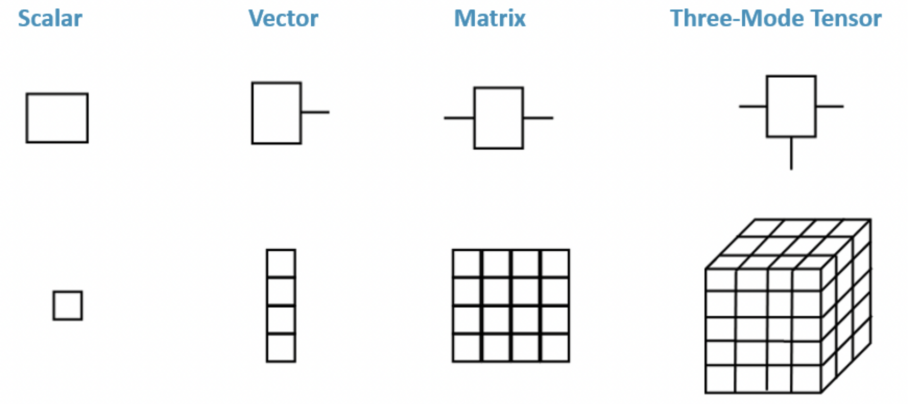


Figure 3.1: Tensor network diagrams of scalars, vectors, matrices, and three-mode tensors [Srinivasan and Adhikary, 2021]

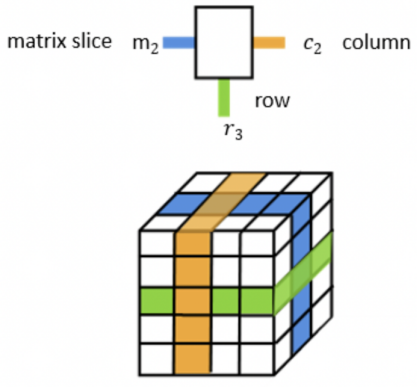


Figure 3.2: Indexing into a single element of a 3-mode tensor by fixing each mode [Srinivasan and Adhikary, 2021]

Tensor networks are a well-studied method to approximately and efficiently model certain kinds of high-dimensional systems. They are of particular interest for modeling quantum systems which scale exponentially in the number of qubits: an N qubit sys-

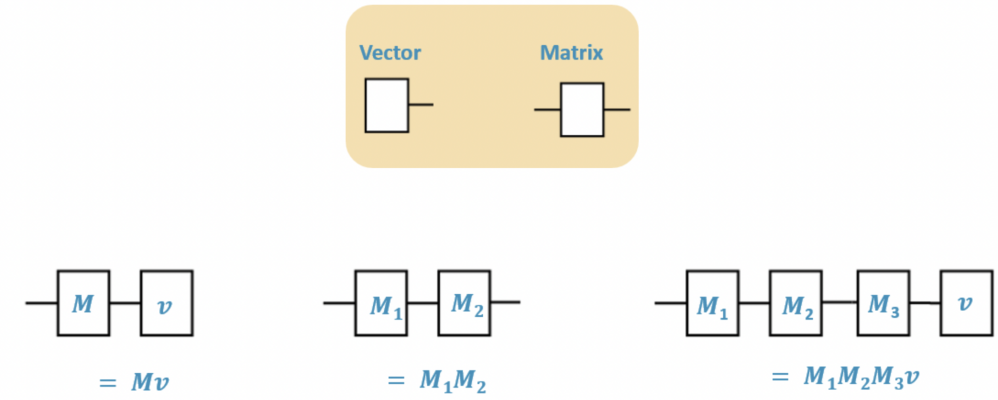


Figure 3.3: Matrix-vector product, matrix-matrix poduct, and matrix-matrix-matrix-vector product in tensor network diagram [Srinivasan and Adhikary, 2021]

tem has a 2^N dimensional state (consisting of probability amplitudes over all N-length bitstrings), and the density matrix has dimension $2^N \times 2^N$. It is not computationally feasible to exactly model these systems, and tensor networks allow us to tractably simulate such large systems by making assumptions on the structure of the high-dimensional tensor. Thus, tensor networks are applied typically when it is helpful to decompose a high-dimensional tensor [Stoudenmire and Schwab, 2016, Mahajan et al., 2021] or solving quantum problems [Cong et al., 2019], potentially as a precursor to implementing such models on quantum hardware [Huggins et al., 2019]. Three of the most common kinds of tensor network decompositions are the 1-D chain of matrix product states (MPS), 2-D grid Projected Entangled Pair States (PEPS), and the tree-structured multi-scale entanglement renormalization ansatz (MERA). We borrow the figure from Biamonte and Bergholm [2017] illustrating these common decompositions in Figure 3.4. Our work (which we present in this chapter) primarily focuses on MPS and the related family of 'flat' tensor network decompositions.

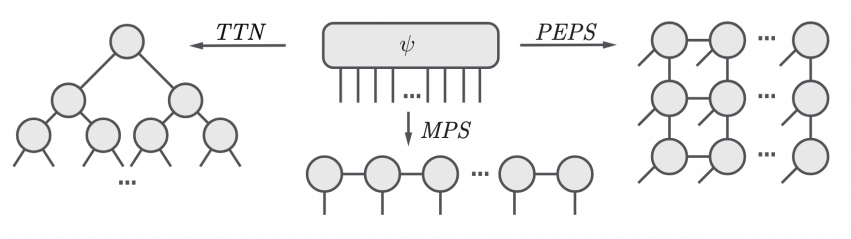


Figure 3.4: Figure from Biamonte and Bergholm [2017]; Three common tensor network structures for modeling high-dimensional tensors

3.2 Linear Stochastic Processes, Tensor Networks, Weighted Automata

In this section, we present our work from Srinivasan et al. [2020], showing how various popular quantum tensor network models have equivalent representations in the stochastic processes and weighted automata literature in the limit of infinitely long sequences. In particular, we will be relating tensor networks to our work on *time-independent* PSRs, HMMs, and HQMMs, so we consider *uniform* tensor networks that have identical tensors at each node. This will allow us to establish expressiveness results for these various tensor network models by borrowing our expressiveness results from Chapter 2. We will also relate these tensor network models to the weighted automata literature for completeness. Our work is most similar to that Glasser et al. [2019], who study tensor network expressiveness for the *non-uniform* case. Our primary results (as illustrated in Figure 3.5) are:

- Uniform Matrix product states (uMPS) are equivalent to PSRs and stochastic weighted automata, when taken in the *non-terminating limit* (where we evaluate probabilities sufficiently away from the end of a sequence).

- Using the known equivalence between uMPS with non-negative parameters, HMMs, and probabilistic automata, we show that another subclass of uMPS called Born machines (BM) [Han et al., 2018] is equivalent to NOOMs [Zhao and Jaeger, 2010] and quadratic weighted automata (QWA) [Bailly, 2011].

- We analyze a broadly expressive subclass of uMPS known as locally purified states

(LPS), and demonstrate its equivalence to hidden quantum Markov models (HQMMs).

We thus develop a unifying perspective on a wide range of models coming from different communities, providing a rigorous characterization of how they are related to one another.

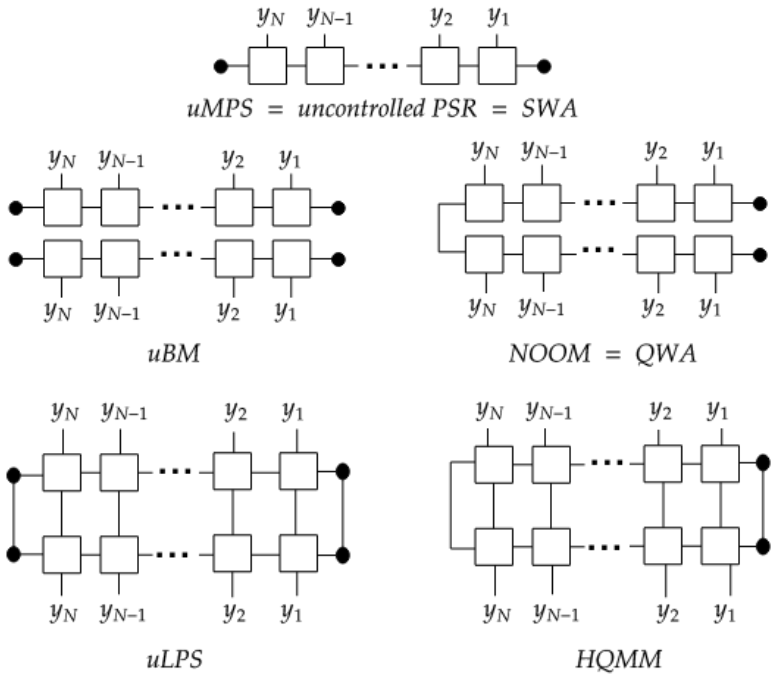


Figure 3.5: Relevant Tensor Network Diagrams: White squares correspond to tensors with as many modes as lines emanating from them, and black end dots indicate boundary vectors. A connecting line represents contraction along a mode, while adjacent tensors without connecting lines are multiplied together via the Kronecker product. Connecting lines (without black dots) at the boundaries represent the application of the identity.

3.2.1 Background

Matrix product states (MPS) were first developed by the physics community as compact representations of often intractable wave functions of complex quantum systems [Perez-Garcia et al., 2006, Klümper et al., 1993, Fannes et al., 1992], in parallel with the equivalent tensor-train decomposition [Oseledets, 2011] developed in applied mathematics for high-order tensors. These tensor network models have been gaining popularity in machine learning, especially as means of compressing highly-parameterized models [Novikov et al., 2015, Garipov et al., 2016, Yu et al., 2017, Novikov et al., 2014]. There has also been

recent interest in directly connecting ideas and methods from quantum tensor networks to machine learning [Stoudenmire and Schwab, 2016, Han et al., 2018, Guo et al., 2018, Huggins et al., 2019]. In particular, tensor networks have been used for probabilistic modeling as parameterizations of joint probability tensors [Glasser et al., 2019, Miller et al., 2021, Stokes and Terilla, 2019]. At the same time, there exist many variants of tensor network models related to MPS that can be used for probabilistic modeling. Glasser et al. [2019] recently provided a thorough investigation of the relative expressiveness of various tensor networks for the *non-uniform* case (where cores in the tensor decomposition need not be identical).

While there has been some investigation connecting tensor networks to various stochastic process models [Kliesch et al., 2014], there had not previously been as thorough an investigation for the *uniform* case. As it turns out, the problem of modeling observation sequences that we explored in Chapter 2 is nearly identical to the problem of efficiently modeling joint probability tensor. We demonstrate that PSRs, HMMs, and HQMMs can be viewed as equivalent to variants of the well-known matrix product state tensor network, and draw on our results in Chapter 2 to derive a collection of results analyzing the relationships in expressiveness between uniform MPS and their various subclasses.

Motivation The uniform case is important to examine for a number of reasons. The inherent weight sharing in uniform tensor networks leads to particularly compact models, especially when learning from highly structured data. This compactness becomes especially useful when we consider physical implementations of tensor network models in quantum circuits. For instance, Glasser et al. [2019] draw an equivalence between local quantum circuits and tensor networks; network parameters define gates that can be implemented on a quantum computer for probabilistic modeling. Uniform networks have fewer parameters, corresponding to a smaller set of quantum gates and greater ease of implementation on resource-constrained near-term quantum computers. Despite the many useful properties of uniformity, the tensor-network literature has tended to focus more on non-uniform models. We fill this gap by developing expressiveness relationships for

uniform variants.

We expect that the connections we establish will also open the door for results and methods in one area to be applied in another. For instance, the learning algorithm for locally purified states (LPS) employed in Glasser et al. [2019] does not preserve uniformity of the model across time-steps, or enforce normalization constraints on learned operators. By establishing an equivalence between uniform LPS and HQMMs, the HQMM learning algorithm from Section 2.5.6 based on optimization over the Stiefel manifold, can be adapted to learn uniform LPS *while enforcing all appropriate constraints*. Similarly, spectral algorithms that have been developed for HMMs and PSRs [Hsu et al., 2012, Siddiqi et al., 2010, Bailly et al., 2009] (see Section 2.3.4) could be adapted to learn uniform LPS and uniform MPS models. These spectral algorithms typically come with consistency guarantees, along with rigorous bounds on sample complexity. Such formal guarantees are less common in tensor network methods, such as variants of alternating least squares [Oseledets, 2011] or density matrix renormalization group methods [White, 1992]. On the other hand, tensor network algorithms tend to be better suited for very high-dimensional data; presenting an opportunity to adapt them to scale up algorithms for stochastic process models.

Notation Our notation is largely similar to Chapter 2. We use bold-face for matrix and tensor operators (e.g. $\mathbf{A}$), arrows over symbols to denote vectors (e.g. $\vec{x}$), and plain non-bold symbols for scalars. The vector-arrows are also used to indicate vectorization (column-first convention) of matrices. We frequently make use of the ones matrix $\mathbf{1}$ (filled with 1s) and the identity matrix $\mathbb{I}$, as well as their vectorizations $\vec{\mathbf{1}}$ and $\vec{\mathbb{I}}$. We use overhead bars to denote complex conjugates (e.g. $\bar{\mathbf{A}}$) and $\dagger$ for the conjugate transpose ($\bar{\mathbf{A}}^T = \mathbf{A}^\dagger$). Finally, $\mathrm{tr}(\cdot)$ denotes the trace operation applied to matrices, and $\otimes$ denotes the Kronecker product.

3.2.2 Uniform Matrix Product States and PSRs

Matrix Product States While MPS are typically used to tractably mode N *spatially separated* quantum systems, we simply adapt it to mode a sequence of N observations.

When each outcome can take d_i values, the joint probability of any particular sequence $y_1, \ldots, y_N$ can be written using the following tensor-train decomposition, which gives an MPS:

$$\begin{aligned} P(y_1, \ldots, y_N) &\propto \text{MPS}_{y_1,\ldots,y_N} \\ &= \mathbf{A}^{[N],y_N} \mathbf{A}^{[N-1],y_{N-1}} \ \ldots \ \mathbf{A}^{[2],y_2} \mathbf{A}^{[1],y_1} \end{aligned} \tag{3.1}$$

where each $\mathbf{A}^{[i]}$ is a three-mode tensor *core* of the MPS containing d_i slices, with the matrix slice associated with outcome Y_i denoted by $\mathbf{A}^{[i],y_i}$. The above joint probability distribution must be normalized through a partition function. A common strategy to efficiently compute this partition function is to convert the tensor chain into a canonical form wherein expensive tensor contractions reduce to the identity. Each matrix slice $\mathbf{A}^{[i],y_i}$ is a $D_{i+1} \times D_i$ matrix, and the conventional choice (which we use in this paper) of *open boundary conditions*[1] is to set $D_0 = D_N = 1$ (i.e. $\mathbf{A}^{[1],y_1}$ and $\mathbf{A}^{[N],y_N}$ are column and row vectors respectively). MPS with open boundaries are equivalent to tensor train (TT) decompositions, and we will define them over the complex field, a choice common in quantum physics and tensor network settings. The maximal value of $D = \max_k D_k$ is also called the **bond-dimension** or the TT-rank [Glasser et al., 2019] of the MPS. [2] For fixed dynamics, this will lead the MPS cores $\mathbf{A}^{[i]}$ to be identical.

Uniform Matrix Product States We will focus on the "uniform" case of identical cores, i.e., a uniform MPS or uMPS. uMPS models were first developed in the quantum physics community [Perez-Garcia et al., 2006, Vanderstraeten et al., 2019], although employing a different probabilistic correspondence (that of Born machines as discussed later) than described below. As we will discuss, this corresponds naturally to Markovian dynam-

[1] An alternate choice, *periodic boundary conditions*, sets $D_0 = D_N \geq 1$ and uses a trace operation to evaluate the product of matrices in Equation 3.1. MPS with periodic boundaries are equivalent to the tensor ring decomposition [Mickelin and Karaman, 2018].

[2] Operational characterizations of the bond dimension have been developed in quantum physics, in terms of entanglement [Eisert et al., 2010] or the state space dimension of recurrent many-body dynamics which generate the associated wavefunction [Schoen et al., 2005].

ical systems; the notion of *past being independent of future given the present* is encoded by the tensor train structure where each core only has two neighbours. While an MPS is inherently defined with respect to a fixed sequence length, a uMPS can be applied to sequences of arbitrary fixed or infinite length [Cirac and Sierra, 2010]. As there should be no distinction between the cores at different time steps in a uMPS, a natural representation is to fix two boundary vectors (suggestively named $\vec{\sigma},\ \vec{\rho}_0$), leading to the following decomposition of the joint probability:

$$\begin{aligned} P(y_1, \ldots, y_N) &= \text{uMPS}_{y_1,\ldots,y_N} \\ &= \vec{\sigma}^\dagger \mathbf{A}^{y_N} \mathbf{A}^{y_{N-1}} \mathbf{A}^{y_2} \mathbf{A}^{y_1} \vec{\rho}_0 \end{aligned} \tag{3.2}$$

Non-terminating uMPS To explore connections with arbitrary-length PSRs and Weighted Finite Automata (WFAs), we will particularly focus on the *non-terminating limit*. Consider that if we wished to compute the probability of some subsequence from $t = 1, \ldots, T$ of the N-length uMPS ($T < N$), we could marginalize over observations $y_{T+1}, \ldots, y_N$. The non-terminating limit is essentially when we consider the uMPS to be infinitely long, i.e., we compute the probability of the subsequence in the limit as $N \to \infty$.

Definition 11 (Non-terminating uMPS)**.** *A non-terminating uMPS is an infinitely long uMPS where we can compute the probability of any sequence $P(y_1, \ldots, y_T)$ of length T by marginalizing over infinitely many future observations, i.e.*

$$P(y_1, \ldots, y_T) = \lim_{N\to\infty} \sum_{y_N} \cdots \sum_{y_{T+1}} P(y_1, \ldots, y_T, y_{T+1}, \ldots, y_N) \tag{3.3}$$

This is a natural approach to modeling arbitrary length sequences with Markovian dynamics; intuitively, if given an identical set of tensor cores at each time step, the probability of a sequence should not depend on how far it is from the 'end' of the sequence. Similar notions are routinely used in machine learning and physics. In machine learning, it is common to discard the first few entries of sequences as "burn-in" to allow systems to

reach their stationary distribution. In our case, the burn-in is being applied to the end of the sequence. The non-terminating limit is also similar to the "thermodynamic limit" employed in many-body physics, which marginalizes over an infinite number of future *and* past observations [Vanderstraeten et al., 2019]. Such limits reflect the behavior seen in the interior of large systems, and avoid more complicated phenomena which arise near the beginning or end of sequences.

Non-terminating uMPS = PSR

While connections between MPS and hidden Markov models (HMMs) have been widely noted, the exact correspondence between non-terminating uMPS models and PSRs (as well as stochastic weighted finite automata (stochastic WFA) [Balle et al., 2014a]) had not been previously explored.

We begin by noting that by Condition 2 of Definition 5 of PSRs, the observable operators $\{\boldsymbol{\tau}_y\}$ are constrained to have normalized marginals over observations $\vec{\sigma}^\dagger \sum_y \boldsymbol{\tau}_y = \vec{\sigma}^\dagger$, i.e., $\vec{\sigma}^\dagger$ is a fixed point of the *'transfer operator'* $\sum_y \boldsymbol{\tau}_y$. The probability of arbitrary length sequences $y_1, \ldots, y_T \in \mathcal{O}^T$ is computed as $\vec{\sigma}^\dagger \boldsymbol{\tau}_{y_T} \ldots \boldsymbol{\tau}_{y_1} \vec{x}_0$, which should be non-negative for any sequence (by Condition 3). This joint probability computation is identical to Equation 3.2, where evaluation functional $\vec{\sigma}$ and the initial state $\vec{x}_0$ are analogous to the left and right boundary vectors of the uMPS, and the *observable operators* $\boldsymbol{\tau}_y$ correspond to the matrix slices $\mathbf{A}^y$. In this sense, both uMPS and PSRs define tensor-train decompositions of joint distributions for a given fixed number of observations T. The only difference is that a uMPS does not require its evaluation functional to be the fixed point of its transfer operator. However, as we will now see, any arbitrary uMPS evaluation functional will eventually converge to the fixed point of its transfer operator in the non-terminating limit. The fixed point then becomes the *effective* evaluation functional of the uMPS in this limit.

Since PSRs were formulated with dynamical systems in mind, we typically consider sequences of *arbitrary* length, whose probabilities are determined via a hidden state which evolves under a time-invariant update rule: the state update conditioned on an observation

y_t is computed as $\vec{x}_t = \frac{\boldsymbol{\tau}_{y_t}\vec{x}_{t-1}}{\vec{\sigma}^T\boldsymbol{\tau}_{y_t}\vec{x}_{t-1}}$ and the probability of an observation y_t is $P(y_t|\vec{x}_t) = \vec{\sigma}^T\boldsymbol{\tau}_{y_t}\vec{x}_t$. This allows us to deal more flexibly with arbitrary length sequences as compared to a generic uMPS. This flexibility for arbitrary-length sequences is precisely why we consider non-terminating uMPS: we can compute the conditional probability of a sequence $P(y_t|y_{1:t-1})$ by marginalizing over all possible future observations with a relatively simple step:

$$P(y_t|y_{1:t-1}) = \frac{\sum_{y_N,\ldots,y_{t+1}} P(y_N,\ldots,y_{t+1},y_t,y_{t-1},\ldots y_1)}{\sum_{y_N,\ldots,y_t} P(y_N,\ldots,y_t,y_{t-1},\ldots,y_1)} = \frac{\vec{\sigma}^\dagger\left(\sum_y \boldsymbol{\tau}_y\right)^{N-(t+1)}\boldsymbol{\tau}_{y_t}\ldots\boldsymbol{\tau}_1\vec{\rho}_0}{\vec{\sigma}^\dagger\left(\sum_y \boldsymbol{\tau}_y\right)^{N-t}\boldsymbol{\tau}_{y_{t-1}}\ldots\boldsymbol{\tau}_1\vec{\rho}_0} \tag{3.4}$$

Thus the *effective* evaluation functional $\vec{\sigma}^\dagger\left(\sum_y \boldsymbol{\tau}_y\right)^{N-t}$ is a function of time and so different at every time step in general. However, if the transfer operator $\boldsymbol{\tau} = \sum_y \boldsymbol{\tau}_y$ has some fixed point $\vec{\sigma}_*^\dagger$, i.e., $\vec{\sigma}_*^\dagger\boldsymbol{\tau} = \vec{\sigma}_*^\dagger$, then the effective evaluation functional at timestep t (which is $N-t$ steps away from the left boundary of the uMPS) will eventually converge to $\vec{\sigma}_*$ given a long enough sequence.[3] So, as long as we remain sufficiently far from the end of a sequence, the particular choice of the the left boundary vector does not matter.

Given that the non-terminating limit effectively replaces the uMPS evaluation functional $\vec{\sigma}$ by the fixed point $\vec{\sigma}_*$, consider what happens if we require $\vec{\sigma} = \vec{\sigma}_*$ to begin with, as is the case for PSRs. In this case, our effective evaluation functional remains independent of t, permitting a simple recursive state update rule that does not require fixing a prior sequence length or marginalizing over future observations. It is in this sense that a non-terminating uMPS is strictly equivalent to a PSR, and we state this formally below.

Theorem 7. *Non-terminating uniform matrix product states are equivalent to uncontrolled predictive state representations.*

Proof. Consider a uMPS $\{\vec{\sigma}, \{\boldsymbol{\tau}_y\}_y, \vec{\rho}_0\}$ representing a joint probability distribution and

[3] In fact, the effective evaluation functional will converge at an exponential rate towards the fixed point, so that $\|\vec{\sigma}_t - \vec{\sigma}_*\|^2 = \mathcal{O}(\exp\frac{N-t}{\xi})$, with a "correlation length" $\xi \simeq (1-|\lambda_2|/|\lambda_1|)^{-1}$ set by the ratio of the largest and second largest eigenvalues of the transfer operator [Orús, 2014]. Transfer operators with non-degenerate spectra can always be rescaled to have a unique fixed point, while those with degenerate spectra form a measure zero subset.

defined over sequences of length N and define the transfer operator $\boldsymbol{\tau} := \sum_y \boldsymbol{\tau}_y$. Say we want to compute the probability of observing y_t at time t, conditioned on past observations $y_{1:t-1}$. This requires us to not only condition on the past observations, but also marginalize over all possible future sequences $y_{t:N}$ (for example, see Miller et al. [2021] for the case of uBMs). The probability is calculated as follows:

$$\begin{aligned} P(y_t \mid y_{1:t}) &= \frac{\vec{\sigma}^\dagger \boldsymbol{\tau}^{N-t} \boldsymbol{\tau}_{y_t} \left(\boldsymbol{\tau}_{y_{t-1}} \cdots \boldsymbol{\tau}_{y_1} \vec{\rho}_0 \right)}{\vec{\sigma}^\dagger \boldsymbol{\tau}^{N-t} \boldsymbol{\tau} \left(\boldsymbol{\tau}_{y_{t-1}} \cdots \boldsymbol{\tau}_{y_1} \vec{\rho}_0 \right)} \\ &= \frac{\vec{\sigma}_t^\dagger \boldsymbol{\tau}_{y_t} \left(\boldsymbol{\tau}_{y_{t-1}} \cdots \boldsymbol{\tau}_{y_1} \vec{\rho}_0 \right)}{\vec{\sigma}_t^\dagger \boldsymbol{\tau} \left(\boldsymbol{\tau}_{y_{t-1}} \cdots \boldsymbol{\tau}_{y_1} \vec{\rho}_0 \right)} \end{aligned} \tag{3.5}$$

Here, we have defined the *effective* evaluation functional $\vec{\sigma}_t = \left(\boldsymbol{\tau}^{N-t}\right)^\dagger \vec{\sigma}$ at time t. Intuitively, this represents the evaluation functional after having marginalized over all possibilities for the remaining $N - t$ time steps. We perform this marginalization via the transfer operator $\boldsymbol{\tau}^\dagger$. As $N \to \infty$ for fixed t, the trajectory of the effective evaluation functional will be strongly determined by the spectral properties of $\boldsymbol{\tau}$. Here, we restrict ourselves to transfer operators where the magnitudes of the two largest eigenvalues are distinct. Note that this is not a strong requirement, given that matrices with degenerate spectra form a measure zero subset of general square matrices[4].

Now if the top eigenvalue of $\boldsymbol{\tau}^\dagger$ is $\lambda_* = 1$, $\vec{\sigma}_t$ will eventually converge to $\vec{\sigma}_*$, the corresponding fixed point of $\boldsymbol{\tau}^\dagger$, owing to the second largest eigenvalue of $\boldsymbol{\tau}^\dagger$ satisfying $|\lambda_2| < 1$. The probability computation in Equation 3.5 then reduces to that of a PSR with evaluation functional $\vec{\sigma}_*$ (and satisfies all conditions in the PSR definition with $\vec{\sigma} = \vec{\sigma}_*$. Note that in a PSRs, the $\vec{\sigma}$ is chosen precisely to be the fixed point of the adjoint transfer operator via the normalization requirement $\vec{\sigma}^\dagger \boldsymbol{\tau} = \vec{\sigma}^\dagger$, forcing $\lambda_* = 1$.

If $\lambda_* \neq 1$ for a uMPS model, we can simply rescale our matrices $\boldsymbol{\tau}$ to obtain a prop-

[4]uMPS which violate this non-degeneracy condition are associated with "(anti-)ferromagnetic order" in quantum-many body physics, and can produce *periodic* behavior in the limit of non-terminating sequences [Cuevas et al., 2017]. Although we don't give the full details here, such degenerate uMPS can still be converted to equivalent PSR by redefining the observation space to consist of k-tuples of adjacent observations, for k the periodicity of the uMPS. This procedure is also called 'blocking' in the condensed-matter literature [Cirac et al., 2017].

erly normalized transfer operator, by replacing $\boldsymbol{\tau}_y$ with $\boldsymbol{\tau}_y/\lambda_*$. Making this substitution in Equation 3.5, we see that the numerator and denominator are rescaled by the same constant λ_*^{t-N}, leaving the overall probability distribution unchanged. As before, this new model produces exactly the same probabilities as a PSR with the evaluation functional $\vec{\sigma}_*$, for $\vec{\sigma}_*$ the fixed point of $\boldsymbol{\tau}^\dagger$. □

If we do not consider the non-terminating limit of a uMPS, the subsequence length and boundary choice will affect the computed probability. Then, we technically only have an equivalence with PSRs for a fixed sequence length (when the evaluation functional is a fixed point of the transfer operator) and no notion of recursive state update. Lastly, note that the *bond dimension* of the MPS is equal on the state dimension of the PSR, which is itself determined by the rank of the Hankel matrix of the linear stochastic process.

Stochastic Weighted Automata A weighted automaton (WA) is a tuple $(\mathbb{C}^n,\ \vec{\sigma},\ \{\boldsymbol{\tau}_y\}_{y\in\mathcal{O}},\ \vec{x}_0)$ which computes a function $f(y_1,\ldots,y_T) = \vec{\sigma}^\dagger \boldsymbol{\tau}_{y_T}\ldots\boldsymbol{\tau}_{y_1}\vec{x}_0$. In contrast with PSRs, no constraints are enforced on the weights of a weighted automaton in general. A weighted automaton is *stochastic* if it computes a probability distribution. These models constitute another class of models equivalent to PSRs and uMPS, and represent probability distributions over sequences of symbols from an alphabet $\mathcal{O}$. As discussed earlier, we focus on models with a finite number of alphabets. It is worth mentioning that the semantics of the probabilities computed by PSRs and stochastic WAs can differ: while PSRs typically maintain a recursive state and are used to compute the probability of a given sequence conditioned on some past sequence, stochastic WAs are often used to compute the joint distributions over the set of all possible finite length sequences (just as in uMPS). We refer the reader to Thon and Jaeger [2015] for a unifying perspective.

The Negative Probability Problem for MPS A similar issue to the negative probability problem from PSRs (Section 2.3.1) arises in the many-body physics setting as well; [Kliesch et al., 2014] show that the question of whether a general *matrix product operator* (commonly used as the tensor decomposition of density matrices) describes a non-negative

quantum density operator is also undecidable. In our context, the exact sense in which the negative probability problem applies is that it is undecidable whether non-terminating uMPS will generate a negative probability.

Model Expressiveness When discussing the expressiveness of a tensor network structure $\mathcal{A}$ relative to a uMPS $\mathcal{M}$, if the bond dimension of $\mathcal{A}$ must *grow* with sequence length in order to represent the uMPS tensor $\mathcal{M}$ of the same length, we say $\mathcal{A}$ is less expressive than the uMPS $\mathcal{M}$. This notion captures the idea that if the state dimension of a model for linear stochastic processes (analogous to bond dimension of uMPS) must grow with the length of the distribution it is trying to model, it could need an arbitrarily large state space for arbitrarily long sequences, and cannot be as expressive a model with a finite-dimensional state space.

Non-Negative uMPS and HMMs

Matrix product states have often been compared to hidden Markov models [Kliesch et al., 2014, Critch et al., 2014], but the actual exact correspndence lies between HMMs and non-terminating non-negative[5] uMPS. Specifically, HMMs are a special case of uMPS when the left boundary $\vec{\sigma}^\dagger = \mathbf{1}^T$, the right boundary $\vec{\rho}_0 = \vec{x}_0$, and the tensor core slice $\mathbf{A}^y = \mathbf{T}_y$.

Probabilistic Automata Non-negative uMPS are also equivalent to probabilistic automata from formal language theory [Denis and Esposito, 2008, Section 4.2], which are in essence weighted automata where transition matrices need to satisfy stochasticity constraints. The strict equivalence between probabilistic automata and HMMs is proved in Dupont et al. [2005, Proposition 8] (see also Section 2.2 in Balle et al. [2014b]).

3.2.3 Uniform Born Machines and NOOMs

Born machines (BMs) [Han et al., 2018] are a popular class of quantum tensor networks that model probability densities as the absolute-square of the outputs of a tensor-train

[5] We refer to uMPS where all tensor cores and boundary vectors are non-negative as *non-negative uMPS*

decomposition, and hence always output valid probabilities. They can be thought of as using matrix product states to represent a joint probability amplitude tensor, i.e., a square root of the probability tensor. The joint probability of N discrete random variables $\{Y_i\}_{i=1}^N$ given by a uniform Born machines (uBMs) [Miller et al., 2021] (see Figure 3.5) is computed as follows (with boundary vectors $\vec{\alpha}$ and $\vec{\omega}_0$ sandwiching a sequence of identical cores $\mathbf{A}$):

$$P(y_1, \ldots, y_N) = \text{uBM}_{y_1,\ldots,y_N} = \left| \vec{\alpha}^{\,\dagger} \mathbf{A}^{y_N} \ldots \ \mathbf{A}^{y_1} \vec{\omega}_0 \right|^2 \tag{3.6}$$

We can re-write this decomposition showing uBMs to be special kinds of uMPS (exactly as we did in Equation 2.10):

$$\begin{aligned} \text{uBM}_{y_1,\ldots,y_N} &= \left| \vec{\alpha}^{\,\dagger} \mathbf{A}^{y_N} \ldots \mathbf{A}^{y_1} \vec{\omega}_0 \right|^2 \\ &= \vec{\alpha}^{\,\dagger} \mathbf{A}^{y_N} \ldots \mathbf{A}^{y_1} \vec{\omega}_0 \ \vec{\omega}_0^{\dagger} (\mathbf{A}^{y_1})^{\dagger} \ldots (\mathbf{A}^{y_N})^{\dagger} \vec{\alpha} \\ &= \vec{\sigma}^{\,\dagger} \ \boldsymbol{\tau}_{y_N} \ \ldots \ \boldsymbol{\tau}_{y_1} \ \vec{\rho}_0 \end{aligned} \tag{3.7}$$

where $\boldsymbol{\tau}_y = \overline{\mathbf{A}^y} \otimes \mathbf{A}^y$, $\vec{\rho}_0 = \overline{\vec{\omega}_0} \otimes \vec{\omega}_0$, and $\vec{\sigma} = \overline{\vec{\alpha}} \otimes \vec{\alpha}$. This makes it clear that uBMs are a special class of MPS, where the tensor cores $\boldsymbol{\tau}_y$ and boundary conditions must satisfy the additional requirement of having unit Schmidt-rank.

Non-terminating uniform BMs = NOOMs

NOOMs bear a striking resemblance to uniform Born machines (uBMs) and the connection has not been previously explored. Both NOOMs and uBMs associate probabilities with quadratic functions of the state vector, with NOOMs directly using the squared 2-norm of the state to determine observation probabilities. While NOOMs were originally defined on the reals, we use a more general definition over complex numbers.

Note that the NOOM joint distribution in Equation 2.10 is *almost* identical to that of uBMs in Equation 3.7, with $\boldsymbol{\tau}_y$ and $\vec{\rho}_0$ having unit Schmidt rank; but the uBM left boundary $\vec{\sigma} = \vec{\alpha} \otimes \vec{\alpha}$ is unit Schmidt rank while the NOOM evaluation functional $\vec{\sigma} = \vec{\mathbb{I}}$

is necessarily full Schmidt rank. So how can we reconcile these nearly identical models? Similar to our approach in Section 3.2.2, we can consider the uBM for an infinitely long sequence where the exact specification of the left boundary / evaluation functional ceases to matter in the non-terminating limit; the effective evaluation functional converges to the fixed point of the transfer operator and we have a notion of recursive state update. Assuming that the uBM transfer operator is a similarity transform away from a trace-preserving quantum channel (i.e., which is normalized by satisfying $\sum_y \boldsymbol{\phi}_y^\dagger \boldsymbol{\phi}_y = \mathbb{I}$), we have that an arbitrary 'left boundary' (with unit Schmidt-rank) of such a uBM will eventually converge to $\vec{\mathbb{I}}$, the NOOM's evaluation functional:

Theorem 8. *Non-terminating uniform Born machines are equivalent to norm observable operator models.*

Proof. Note that uniform Born machines and norm observable operator models differ only in their evaluation functionals, and in the requirement that NOOM operators must be trace-preserving. While uBMs can have an arbitrary Schmidt-rank 1 evaluation functional, NOOMs are restricted to the higher-rank identity evaluation functional $\vec{\mathbb{I}}$. Both models have operators of the form $\boldsymbol{\tau}_y = \boldsymbol{\phi}_y \otimes \boldsymbol{\phi}_y$, which are completely positive with Schmidt-rank 1, but uBM transfer operator need not be trace-preserving. We primarily need to show that an arbitrary uBM is equivalent to one where the transfer operator $\boldsymbol{\tau} = \sum_y \boldsymbol{\tau}_y$ is trace-preserving, which is the same as its adjoint $\boldsymbol{\tau}^\dagger$ being unital, i.e., having the identity $\vec{\mathbb{I}}$ as a fixed point. Then, the same argument used in the proof of Theorem 7 to prove convergence of the effective functional of a uMPS to the fixed point $\vec{\sigma}_*^\dagger$ can be applied here. This has the effect of replacing the original Schmidt-rank 1 functional of the uBM by the identity $\vec{\mathbb{I}}$ in the non-terminating limit, completing the conversion from uBM to NOOM.

If the uBM transfer operator $\boldsymbol{\tau}$ is already trace-preserving then we are done, so assume it is not. We make the generic assumption[6] that the two eigenvalues of $\boldsymbol{\tau}$ with greatest

[6]This is similar to the assumption made in proving Theorem 7, and is valid everywhere outside of a measure-zero subset of uBMs. In the case of degenerate eigenvalues, the conversion from uBMs to NOOMs can still be achieved given a slight redefinition of the observation space to combine together k adjacent observations.

magnitude, λ_* and λ_2, satisfy $|\lambda_*| > |\lambda_2|$. By replacing all operators $\boldsymbol{\phi}_y$ by $\boldsymbol{\phi}_y/\sqrt{\lambda_*}$, we convert the uBM into one whose transfer operator $\boldsymbol{\tau}$ has leading eigenvalue of magnitude 1, a rescaling which leaves the joint probability distributions unchanged (similar to how Equation 3.5 was left unchanged by rescaling operators in the uMPS case). In this case, the quantum Perron-Frobenius theorem [Evans and Høegh-Krohn, 1977] then ensures that $\lambda_* = 1$, and that $\boldsymbol{\tau}^\dagger$ has a unique fixed-point operator $\vec{\sigma}_*$ which is the vectorization of a full-rank (and consequently, invertible) positive definite matrix $\boldsymbol{\sigma}_*$.

A similarity transformation of $\mathbf{S} = \boldsymbol{\sigma}_*^{1/2}$ can then be applied to the uBM matrices, replacing $\boldsymbol{\phi}_y$ with $\boldsymbol{\phi}'_y = \mathbf{S}\boldsymbol{\phi}_y\mathbf{S}^{-1}$. This similarity transformation ensures the new transfer operator $\boldsymbol{\tau}'$ is trace-preserving, as demonstrated by the following:

$$\begin{aligned}
\boldsymbol{\tau}'^\dagger\vec{\mathbb{I}} &= \left(\sum_y \overline{\boldsymbol{\phi}'_y}^\dagger \otimes \boldsymbol{\phi}'^\dagger_y\right)\vec{\mathbb{I}} = \left(\sum_y (\boldsymbol{\sigma}_*^{-1/2})^T \overline{\boldsymbol{\phi}_y}^\dagger (\boldsymbol{\sigma}_*^{1/2})^T \otimes \boldsymbol{\sigma}_*^{-1/2}\boldsymbol{\phi}_y^\dagger\boldsymbol{\sigma}_*^{1/2}\right)\vec{\mathbb{I}} \\
&= \left((\boldsymbol{\sigma}_*^{-1/2})^T \otimes \boldsymbol{\sigma}_*^{-1/2}\right)\left(\sum_y \overline{\boldsymbol{\phi}_y}^\dagger \otimes \boldsymbol{\phi}_y^\dagger\right)\left((\boldsymbol{\sigma}_*^{1/2})^T \otimes \boldsymbol{\sigma}_*^{1/2}\right)\vec{\mathbb{I}} \\
&= \left((\boldsymbol{\sigma}_*^{-1/2})^T \otimes \boldsymbol{\sigma}_*^{-1/2}\right)\boldsymbol{\tau}^\dagger\vec{\sigma}_* \\
&= \left((\boldsymbol{\sigma}_*^{-1/2})^T \otimes \boldsymbol{\sigma}_*^{-1/2}\right)\vec{\sigma}_* \\
&= \vec{\mathbb{I}}
\end{aligned}$$

In the equations above, we have used the facts that (a) $\boldsymbol{\sigma}_*$, $\boldsymbol{\sigma}_*^{1/2}$, and $\boldsymbol{\sigma}_*^{-1/2}$ are Hermitian, so that complex conjugation acts as $\overline{\boldsymbol{\sigma}_*^{1/2}} = (\boldsymbol{\sigma}_*^{1/2})^T$, (b) $\left(\mathbf{Z}^T \otimes \mathbf{X}\right)\vec{Y} = \vec{W}$ with $\mathbf{W} = \mathbf{XYZ}$, for any matrices $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$, and (c) $\boldsymbol{\tau}^\dagger\vec{\sigma}_* = \vec{\sigma}_*$

We have demonstrated that the similarity-transformed uBM now possesses a trace-preserving transfer operator, which by the arguments above ensures it is a valid NOOM in the non-terminating limit. □

Expressiveness of Non-terminating uBMs With the above equivalence, we now turn to the question of how the expressiveness of uBM compares to non-negative uMPS. Glasser et al. [2019] studied the expressiveness of *non-uniform* BMs, showing that there are finite-dimensional non-uniform BMs that cannot be modeled by finite dimensional non-uniform HMMs, and conjecture that the reverse direction is also true. Drawing on

our expressiveness results from Section 2.4, we can conclude that in the uniform case, uBMs are not as expressive as uMPS, and can model certain sequences that non-negative uMPS cannot, but also cannot model certain sequences that non-negative uMPS can.

Quadratic Weighted Automata Finally, we note that quadratic weighted automata (QWA) [Bailly, 2011], developed in the stochastic weighted automata literature, are equivalent to uBM. Bailly [2011] suggest that QWA $\nsubseteq$ HMM and that HMM $\nsubseteq$ QWA, but do not provide a proof. To the best of our knowledge, the proof we provide is the first to formally show the non-equivalence of QWA and HMM.

3.2.4 Uniform Locally Purified States and HQMMs

Locally purified states (LPS) (visualized in Figure 3.5) were proposed as a tensor-network model of discrete multivariate probability distributions inspired from techniques used in the simulation of quantum systems. Glasser et al. [2019] show that these models are not only strictly more expressive than non-uniform HMMs, but also correspond directly to local quantum circuits with ancillary qubits – serving as a guide to design quantum circuits for probabilistic modeling. LPS can be thought of two identical MPS connected by an additional mode – called the "purification dimension" – in each of the MPS tensors. The rank of an LPS, also called its puri-rank, is defined the same way as the bond dimension (or TT-rank) for the MPS[7]. The corresponding uniform LPS defines the unnormalized

[7]As we will see, non-terminating uLPS are equivalent to HQMMs, and the 'puri-rank' is exactly analogous to the number of Kraus operators per observable, i.e., the 'environment dimension' denoted by w in Section 2.5.2

probability mass function over N discrete random variables $\{Y_i\}_{i=1}^N$ as follows:

$$\begin{aligned} P(y_1, \ldots, y_N) &= \text{uLPS}_{y_1,\ldots,y_N} \\ &= \left(\sum_{\beta_L} \overline{\mathbf{K}}^T_{\beta_L,L} \otimes \mathbf{K}^T_{\beta_L,L}\right) \left(\sum_{\beta} \overline{\mathbf{K}}_{\beta,y_N} \otimes \mathbf{K}_{\beta,y_N}\right) \cdots \\ &\cdots \left(\sum_{\beta} \overline{\mathbf{K}}_{\beta,y_1} \otimes \mathbf{K}_{\beta,y_1}\right) \left(\sum_{\beta_R} \overline{\mathbf{K}}_{\beta_R,R} \otimes \mathbf{K}_{\beta_R,R}\right) \\ &= \vec{\sigma}^\dagger \mathbf{L}_{y_1} \cdots \mathbf{L}_{y_N} \vec{\rho}_0 \end{aligned} \tag{3.8}$$

where we have suggestively defined $\vec{\sigma}^\dagger = \left(\sum_{\beta_L} \overline{\mathbf{K}}^T_{\beta_L,L} \otimes \mathbf{K}^T_{\beta_L,L}\right)$, $\mathbf{L}_{y_i} = \left(\sum_{\beta} \overline{\mathbf{K}}_{\beta,y_i} \otimes \mathbf{K}_{\beta,y_i}\right)$, and $\vec{\rho}_0 = \left(\sum_{\beta_R} \overline{\mathbf{K}}_{\beta_R,R} \otimes \mathbf{K}_{\beta_R,R}\right)$.

Non-terminating Uniform LPS are HQMMs Equation 3.8 shows that every HQMM is a uLPS, but we also consider in what sense every uLPS is an HQMM: the transfer operator of arbitrary CP maps with unit spectral radius[8] is a similarity transform away from that of a CP-TP map [Perez-Garcia et al., 2006], so $\vec{\mathbb{I}}$ is related to such a fixed point by such a similarity transform. Thus, every non-terminating uLPS has an equivalent HQMM and allows for an HQMM-style recursive state update. This is the same reasoning behind the equivalence between non-terminating uBMs (with CP maps) and NOOMs (with CP-TP maps). We state this formally below:

Theorem 9. *Non-terminating uniform locally purified states are equivalent to hidden quantum Markov models.*

Proof. We follow the same approach as in Theorem 8. Uniform LPS models and HQMMs differ only in their evaluation functionals, and in the requirement that HQMMs operators are trace-preserving. While uLPSs can have an arbitrary evaluation functional, HQMMs are restricted to the identity evaluation functional $\vec{\mathbb{I}}$. Both models have operators of the form $\boldsymbol{\tau}_y = \sum_y \overline{\mathbf{K}_y} \otimes \mathbf{K}_y$, which are completely positive operators.

[8] This condition is necessary for probability distributions such as Equations 3.7 and 3.8 to be properly normalized.

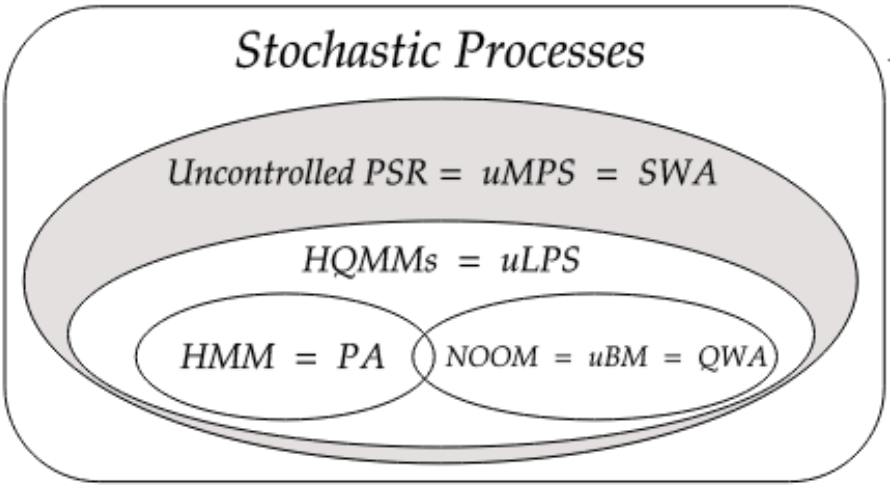


Figure 3.6: Expressiveness Relationships Betweens Models: Subset relationships between stochastic process models, non-terminating uniform quantum tensor networks, and weighted automata. All tensor networks are assumed to be non-terminating. The grey area is potentially empty.

As discussed in Theorem 8, the transfer operator $\boldsymbol{\tau}$ can be rescaled and similarity transformed into one that is trace-preserving. In the limit of non-terminating sequences, the evaluation functional of this transformed model will then converge to $\vec{\mathbb{I}}$, the fixed point of $\boldsymbol{\tau}^\dagger$. Therefore, this similarity transform allows us to map a non-terminating uLPS to an HQMM, proving that non-terminating uLPSs are equivalent to HQMMs. □

We are not aware of any proposals from the weighted automata literature that are analogous to these uLPS/HQMMs.

Expressiveness of HQMMs (uLPS) and PSRs (uMPS) Drawing on our expressiveness results from Section 2.5.4, we can conclude that in the uniform case, uLPS are the most expressive tensor network structure that can model joint probability distributions without suffering from an undecidable NPP. The question of whether there is a 'gap' between uLPS and uMPS persists (Section 2.5.5). The results in Glasser et al. [2019] and De las Cuevas et al. [2013] show that MPS are more expressive than LPS in the *non-uniform* case, but their technique cannot be easily adapted to the uniform case.

3.2.5 Summary

We presented uniform matrix product states and their various subclasses including non-negative uniform MPS, uniform Born machines, and uniform locally purified states. We showed how they relate to previous work in the stochastic processes and weighted automata literature, and drew on our expressiveness results from Chapter 2 to establish the expressiveness of these tensor network models. We also speculate that the connections laid out here may make spectral learning algorithms commonly used for PSRs and weighted automata [Hsu et al., 2012, Balle et al., 2014a, Hefny et al., 2015] suitable for learning uMPS, and an algorithm for optimization on the Stiefel manifold (Section 2.5.6) suitable for learning uLPS *with appropriate constraints*. Future work will involve adapting these algorithms so they can be transferred between the two fields.

3.3 The Difficulty of Formulating CPTP Tensor Networks for Quantum Channels

Here, we look at our previous work [Srinivasan et al., 2021] on applying tensor networks o an entirely quantum problem: the prospects for formulating a (not necessarily uniform) tensor network representation of a quantum channel (called a locally purified density operator or LPDO) that satisfies the completely positive and trace-preservation (CP and TP) constraints (discussed in Section 2.5.1). At a high level, we present the following results from our work:

- We pose and discuss the problem of parameterizing TP LPDOs.
- We show that a naive constraint imposed independently on each LPDO tensor core is sufficient to guarantee the TP property of the LPDO, but is too strict to capture simple 2-qubit gates.
- We provide an alternative characterization of the constraints on each LPDO tensor core, including a result that the tensor cores of TP LPDOs must be orthonormal

2-frames under a metric induced by the other cores.

- We show that there may be insurmountable challenges to constructively defining a TP LPDO.

3.3.1 Background

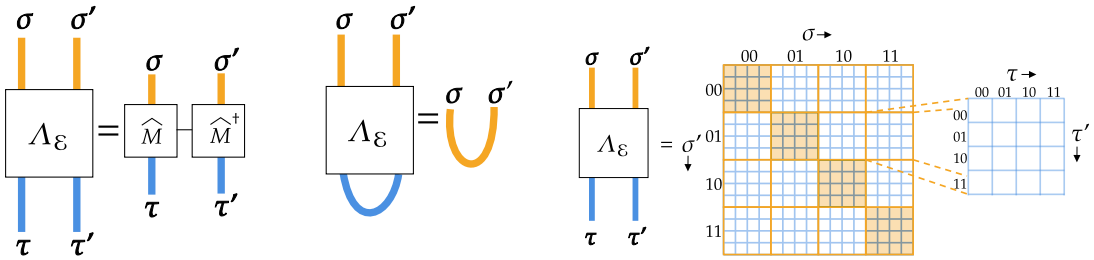
Motivation: Quantum Process Tomography for Large Systems Consider the problem of quantum process tomography, where various prepared quantum states are acted upon by a quantum channel and measured, and the goal is to identify the quantum channel. This may itself be a component of a larger problem like quantum error correction [Kosut et al., 2008]. Cast as a machine learning problem, the goal is to learn to predict the action of the quantum channel given training data of the output measurements corresponding to input prepared states.

We first review some preliminaries on quantum channels and Choi matrices as relevant for this section (also discussed in Section 2.5.1).

Review: Quantum Channels We consider a quantum channel acting on a system of N qubits. Let $\mathcal{H}_A$ denote the 2^N-dimensional Hilbert space of pure input states with basis states $\{|\boldsymbol{\sigma}\rangle\}$ and $\mathcal{H}_B$ denote the 2^N-dimensional Hilbert space of pure output states with basis states $\{|\boldsymbol{\tau}\rangle\}$. Mixed input and output states live in $L(\mathcal{H}_A)$ and $L(\mathcal{H}_B)$, which are spaces of unit-trace, positive semi-definite (PSD) density matrices. A *quantum channel* is a linear map $\mathcal{E} : L(\mathcal{H}_A) \to L(\mathcal{H}_B)$ from density matrices in $L(\mathcal{H}_A)$ to density matrices in $L(\mathcal{H}_B)$. Such a map satisfies two[9] constraints: 1) it must be completely positive (CP), i.e., preserve not only the PSD property of input density matrices, but also the PSD property of input density matrices when coupled with an arbitrary dimensional ancilla system, and 2) it must be trace-preserving (TP), i.e., leave the trace of input density matrices unchanged.

[9]The channel must also preserve the Hermiticity of input density matrices, but this is already guaranteed by the CP constraint.

Review: Choi Matrices A quantum channel $\mathcal{E} : L(\mathcal{H}_A) \to L(\mathcal{H}_B)$ can be represented by a $2^{2N} \times 2^{2N}$ complex-valued *Choi matrix* as $\mathbf{\Lambda}_{\mathcal{E}} = \sum_{\boldsymbol{\sigma},\boldsymbol{\tau}} |\boldsymbol{\tau}\rangle\langle\boldsymbol{\tau}'| \otimes \mathcal{E}\left(|\boldsymbol{\sigma}\rangle\langle\boldsymbol{\sigma}'|\right)$. This permits a simple characterization of the CP-TP constraints: the CP conditions amount to a requirement that $\mathbf{\Lambda}_{\mathcal{E}}$ be a PSD matrix, and the TP condition requires that the partial trace over the output states be identity $\mathrm{tr}_B\left(\mathbf{\Lambda}_{\mathcal{E}}\right) = \mathbb{I}_{2^N}$. These constraints are illustrated in the language of tensor diagrams in Figure 3.7a. We can think of the Choi matrix as having a block structure, where the input basis states $\boldsymbol{\sigma}$, $\boldsymbol{\sigma}'$ pick out a block of the Choi matrix, and output basis states $\boldsymbol{\tau}, \boldsymbol{\tau}'$ pick out a specific entry within the block. This block structure is illustrated in Figure 3.7b.



(a) The complete positivity (**left**) and trace preservation (**right**) constraints on Choi matrices.

(b) Block structure of Choi matrix; indices (σ, σ') pick out matrix blocks and indices (τ, τ') pick out elements within the blocks.

Figure 3.7: Illustrations of Choi matrices and their properties

The Challenge The task of estimating the Choi matrix corresponding to a quantum channel (process tomography) is more complicated than learning quantum states (state tomography)[10]. While the Choi-Jamiolskowski isomorphism [Choi, 1975, Jamiołkowski, 1972] gives a connection between bipartite states and Choi matrices, this is not a true isomorphism [Jiang et al., 2013] since the TP condition requires the input subsystem

[10] As we saw with the correspondence between HQMMs and uniform locally purified states (uLPS), these model a quantum state by virtue of the latent state evolution inducing a distribution over observations (Schoen et al. [2005] give an exact scheme establishing such correspondence for MPS). As it turns out, HQMMs can also easily be viewed as a model for *quantum processes*; all it takes is to view the initial latent state ρ_0 as an input to the system, so different inputs induce different distributions over outputs as with a quantum channel. The challenge however, is that the HQMM/uLPS structure is only compact in modeling the output distribution. The inputs to HQMMs specified by the initial latent state ρ must be the same dimension as the input to the quantum channel ($2^N \times 2^N$), with no compression. After all, HQMMs/uLPS were developed to efficiently represent the distribution over observations for a given *fixed* input. Hence, HQMMs/uLPS are not able to efficiently represent quantum channels.

of the bipartite state corresponding to a Choi matrix to be the identity. In other words, diagonal blocks of the Choi matrix must be unit trace, and off-diagonal blocks must be zero trace. Thus, the problem is not as simple as learning parameters first and renormalizing with a constant at the end (a common approach with MPS), since the diagonal blocks and off-diagonal blocks are (trace) normalized differently. While some have proposed learning a Choi matrix via unconstrained optimization and projecting it onto the set of CP-TP matrices at the end [Knee et al., 2018, Surawy-Stepney et al., 2021], another natural solution is to observe that a Choi matrix is CP-TP if and only if the tensor core $\hat{\mathbf{M}}$ in Figure 3.7a (square root of the Choi matrix $[\mathbf{\Lambda}_{\mathcal{E}}]_{\boldsymbol{\sigma},\boldsymbol{\tau}}^{\boldsymbol{\sigma}',\boldsymbol{\tau}'} = (\hat{\mathbf{M}}_{\mu}^{\boldsymbol{\sigma},\boldsymbol{\tau}})^{\dagger}\hat{\mathbf{M}}_{\mu}^{\boldsymbol{\sigma}',\boldsymbol{\tau}'}$) lives on a Stiefel manifold, and use Riemannian optimization techniques [Luchnikov et al., 2021] such as the Wen-Yin algorithm (see Section 2.5.6) to learn the Choi matrix. This is an important characterization of the TP constraint, so we state it formally below for later use.

Remark 1. *Let* $[\mathbf{\Lambda}_{\mathcal{E}}]_{\boldsymbol{\sigma},\boldsymbol{\tau}}^{\boldsymbol{\sigma}',\boldsymbol{\tau}'} = ([\hat{\mathbf{M}}]_{\mu}^{\boldsymbol{\sigma},\boldsymbol{\tau}})^{\dagger}[\hat{\mathbf{M}}]_{\mu}^{\boldsymbol{\sigma}',\boldsymbol{\tau}'}$ *and* $[\mathbf{M}]_{\boldsymbol{\tau},\mu}^{\boldsymbol{\sigma}}$ *be a reshape of* $[\hat{\mathbf{M}}]_{\mu}^{\boldsymbol{\sigma},\boldsymbol{\tau}}$*. Then, the TP condition on the Choi matrix is equivalently stated as* $\mathbf{M}^{\dagger}\mathbf{M} = \mathbb{I}$*.*

However, the other complication in estimating Choi matrices from data is more severe: since the number of parameters scales exponentially with the number of qubits. Modeling a $2^{2N} \times 2^{2N}$ quantum channel explicitly is simply intractable for large quantum systems. This general problem setting is fertile ground for tensor network solutions, which we consider next. One approach that avoids this curse of dimensionality is factorizing the Choi matrix into a *matrix product operator* (MPO) consisting of a series of N compact tensor cores. However, such an MPO must still represent quantum channels which are CP and TP (see Section 2.5.1), both of which are non-trivial constraints that must be explicitly satisfied in the optimization algorithm.

Locally Purified Density Operators The Locally Purified Density Operator (LPDO) is a tensor network proposal for modeling Choi matrices [Werner et al., 2016, Torlai et al., 2020] that satisfies the CP requirement by design (i.e., the corresponding Choi matrix that it decomposes is PSD). It is quite similar to LPS, except each tensor core has an

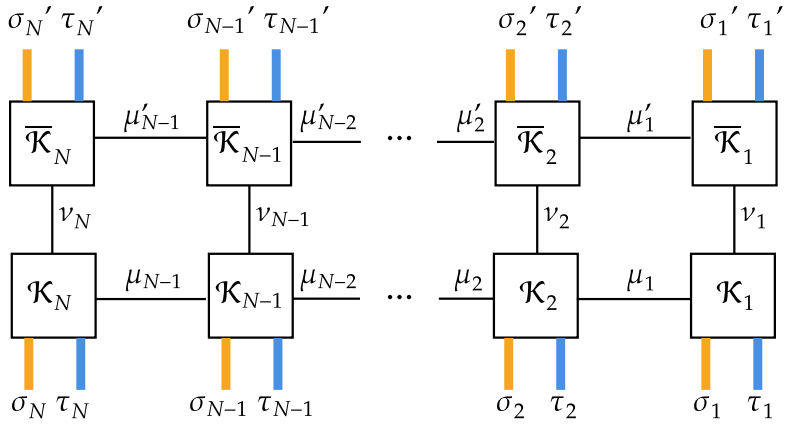


Figure 3.8: Tensor network diagram of a locally purified density operator (LPDO)

additional mode representing 'inputs' to the quantum channel. This makes sense since LPS are ultimately tensor decomposition of a joint probability distribution *vector*, but the LPDO is a decomposition of a Choi *matrix*. The LPDO consists of $2N$ cores (N cores and their conjugates) in the structure shown in Figure 3.8. The j-th tensor core of an LPDO contains the following modes: one input mode σ_j or σ'_j, one output mode τ_j or τ'_j, one purification mode ν_j, and one or two bond modes μ_{j-1}, μ_j or μ'_{j-1}, μ'_j. Denoting the collection of core tensor parameters as θ, the Choi matrix corresponding to such a tensor network is given written as:

$$[\mathbf{\Lambda}_\theta]^{\boldsymbol{\sigma}',\boldsymbol{\tau}'}_{\boldsymbol{\sigma},\boldsymbol{\tau}} = \sum_{\boldsymbol{\mu},\boldsymbol{\mu}',\boldsymbol{\nu}} \prod_{j=1}^{N} [\mathcal{K}_j]^{\nu_j,\mu_j}_{\sigma_j,\tau_j,\mu_{j-1}} [\overline{\mathcal{K}_j}]^{\sigma'_j,\tau'_j,\mu'_j}_{\nu_j,\mu'_{j-1}} \tag{3.9}$$

Leveraging these properties, Torlai et al. [2020] recently proposed a gradient-descent based maximum-likelihood algorithm to learn LPDO approximations of Choi matrices for quantum process tomography. While this procedure yields LPDOs that are always CP, there is no guarantee that the learned LPDO satisfies the TP condition. Torlai et al. [2020] propose adding a TP violation penalty term to their maximum likelihood objective to push the solution towards TP LPDOs, and note that the TP violation shrinks to zero in the limit of infinite data. However, in practice, with limited data the TP violation can still be non-negligible and the learned LPDO can be a non-physical *trace-increasing* channel (see Figure 3.9). There is also no known scheme to project the learned LPDO to

the nearest TP LPDO without reconstructing the full Choi matrix, which would defeat the purpose of using the LPDO decomposition in the first place. For various applications, it may be desirable to parameterize and learn a physically valid LPDO structure that also satisfies the TP constraint. This brings us to the problem we consider in this section: *can we parameterize and optimize* trace-preserving *LPDOs?* If so, we could unlock the ability to employ *physical* LPDO approximations of quantum channels for various tasks including quantum process tomography and quantum error correction, particularly for large quantum systems with many qubits. Unfortunately, we encounter some serious theoretical difficulties in finding such a parameterization.

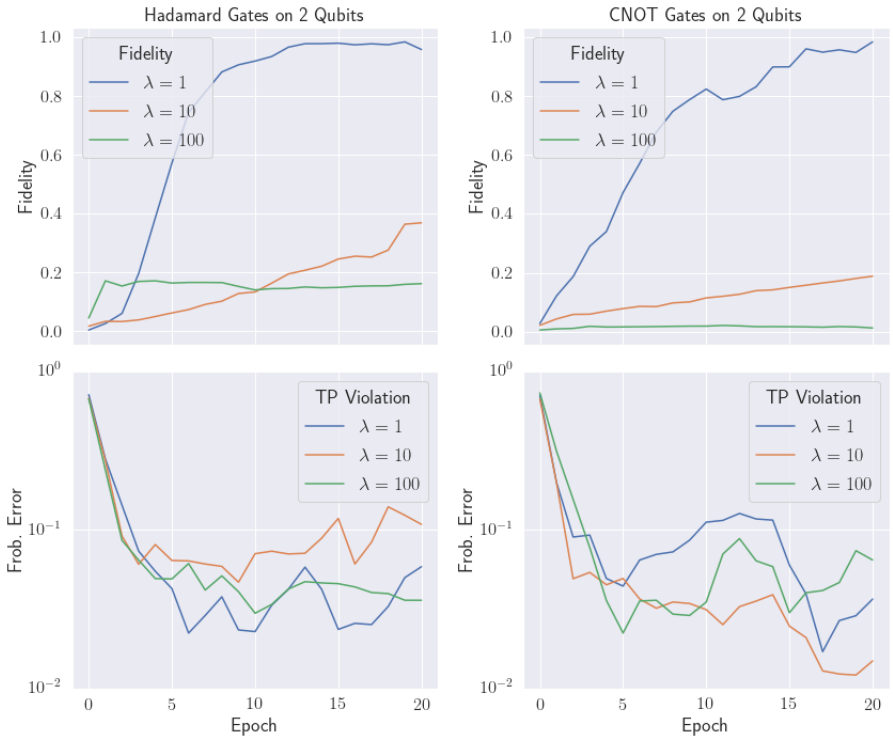


Figure 3.9: Learning simple quantum channels using the method of Torlai et al. [2020] with various penalties λ on TP violation. **Top**: Fidelity of the learned Choi matrix and the true Choi matrix, **Bottom**: TP Violation as Frobenius norm distance of partial trace of learned Choi matrix from identity. Increasing the penalty term does not much affect TP violation error, but does slow convergence.

We first explore a naive constraint we can impose independently on each tensor core that is sufficient to guarantee that the LPDO satisfies a global TP condition. In practice

however, this constraint is too strong and is unable to capture simple 2-qubit unitary gates, let alone more complicated channels. We then present a more involved discussion of the TP condition, and provide a more general characterization of TP LPDOs.

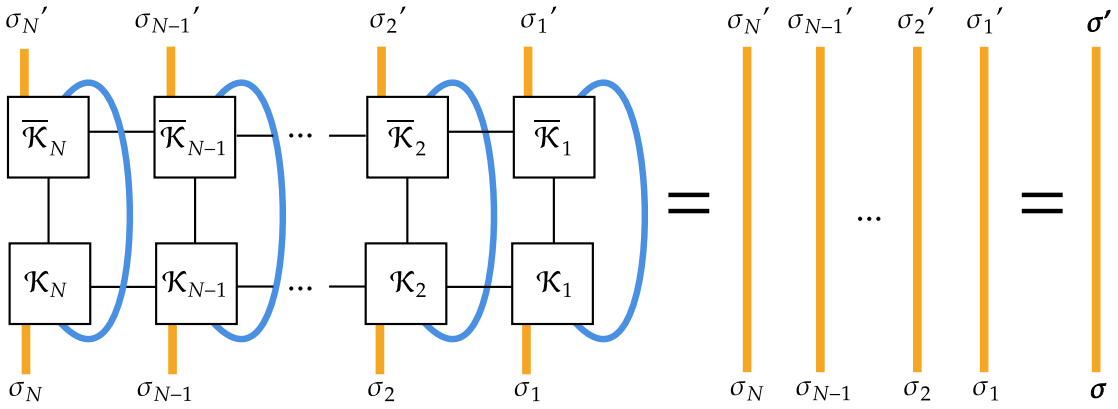


Figure 3.10: The TP constraint on an LPDO representing a quantum channel. The blue connections correspond to taking the partial trace with respect to the output indices, which results in the identity.

3.3.2 A Simple Sufficient Condition for TP LPDOs

One simple approach to ensuring TP LPDOs is to force every core $\mathcal{K}_j, \overline{\mathcal{K}}_j$ to function as a local quantum channel acting on the "outputs" from cores $\mathcal{K}_{j-1}, \overline{\mathcal{K}}_{j-1}$. To do this, we will use the Kraus operator formulation of quantum channels: every CP-TP quantum channel can be written using a set of Kraus operators $\mathcal{E}(\rho) = \sum_i \mathbf{K}_i \rho \mathbf{K}_i^\dagger$, where the set of Kraus operators must satisfy the condition $\sum_i \mathbf{K}_i^\dagger \mathbf{K}_i = \mathbb{I}$. As we noted in Section 2.5.6, vertically stacking a set of Kraus operators produces a matrix that lies on a Stiefel manifold of dimension $2PD \times 2D$, i.e., if $\kappa = [K_1, \ldots, K_W]^T$, then $\sum_i K_i^\dagger K_i = \kappa^\dagger \kappa = \mathbb{I}$ [Kraus, 1971, Gheondea, 2010]. Now, the idea is to interpret every *core* as a set of Kraus operators on a Stiefel manifold (allowing us to use Riemannian optimization techniques on each tensor core), so the application of a sequence of local quantum channels is also a global quantum channel and recovers the global TP condition. To achieve this, we make a slight modification to the LPDO structure by adding an extra mode to the cores at the N-site, and we call this extended LPDO an xLPDO (Figure 3.11). Note that the extended

edge of the xLPDO can be fused with the core's purification dimension to recover the LPDO structure, so it is still in the same class of tensor networks. We state the sufficient condition formally below and give a diagrammatic proof in Figure 3.11.

Lemma 1 (Sufficient Constraint for TP LPDO)**.** *Let κ_j be a flattening of the tensor core $\mathcal{K}_j$ into a matrix with rows associated with $\nu_j\tau_j\mu_j$ and columns with $\sigma_j\mu_{j-1}$. If each κ_j satisfies $\kappa_j^\dagger\kappa_j = \mathbb{I}$ for $1 \leq j \leq N$, , i.e., each κ_j is on the Stiefel manifold, the LPDO satisfies the global TP constraint.*

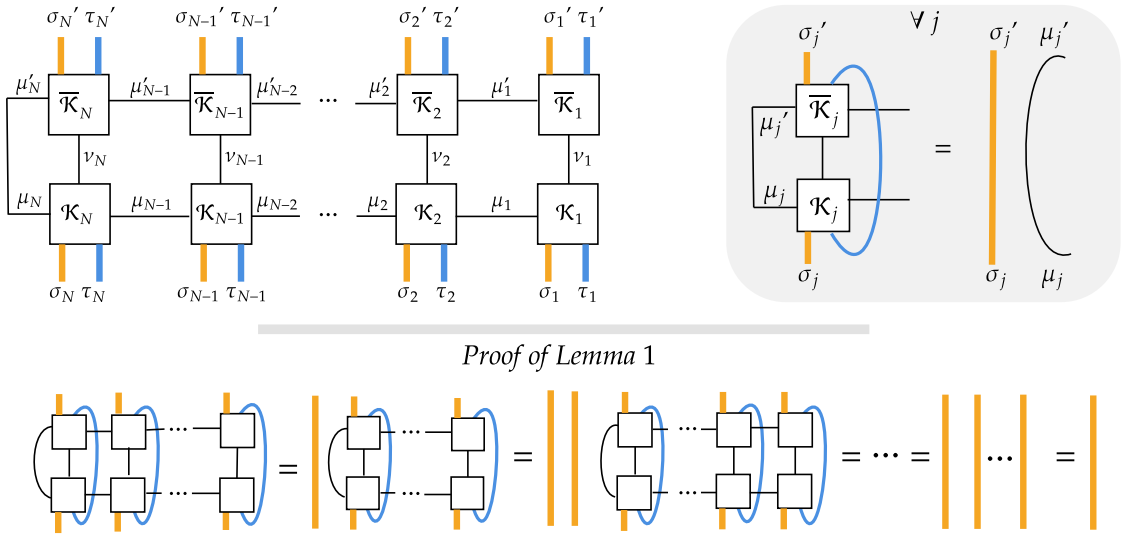


Figure 3.11: **Top:** Illustration of xLPDO (left) and sufficient local Stiefel manifold constraint on each core for TP LPDO (right), **Bottom:** Graphical proof of Lemma 1

Proof. First, we observe that Equation 3.9 can be written as follows:

$$[\mathbf{\Lambda}_\theta]^{\boldsymbol{\tau}',\boldsymbol{\tau}}_{\boldsymbol{\sigma}',\boldsymbol{\sigma}} = \sum_{\boldsymbol{\nu}_j} (\mathbf{K}^{\sigma_1'}_{\tau_1',\nu_1})^\dagger \cdots (\mathbf{K}^{\sigma_N'}_{\tau_N',\nu_N})^\dagger \mathbf{K}^{\sigma_N}_{\tau_N,\nu_N} \cdots \mathbf{K}^{\sigma_1}_{\tau_1,\nu_1} \tag{3.10}$$

The TP requirement on Choi matrices is that $\mathrm{tr}_B(\mathbf{\Lambda}_\theta) = \mathbb{I}$, or equivalently, the trace of diagonal blocks of the Choi matrix is 1 and trace of off-diagonal blocks is 0: $\sum_{\boldsymbol{\tau}=\boldsymbol{\tau}'} [\mathbf{\Lambda}_\theta]^{\boldsymbol{\tau},\boldsymbol{\tau}'}_{\boldsymbol{\sigma}'=\boldsymbol{\sigma}} = 1$ and $\sum_{\boldsymbol{\tau}=\boldsymbol{\tau}'} [\mathbf{\Lambda}_\theta]^{\boldsymbol{\tau},\boldsymbol{\tau}'}_{\boldsymbol{\sigma}'\neq\boldsymbol{\sigma}} = 0$. Observe that:

$$\begin{aligned}\sum_{\boldsymbol{\tau}=\boldsymbol{\tau'}} [\boldsymbol{\Lambda}_\theta]^{\boldsymbol{\tau}',\boldsymbol{\tau}}_{\boldsymbol{\sigma}'=\boldsymbol{\sigma}} &= \sum_{\tau_1,\ldots,\tau_{N-1}} \sum_{\nu_1,\ldots,\nu_{N-1}} (\mathbf{K}^{\sigma_1}_{\tau_1,\nu_1})^\dagger \cdots \left(\sum_{\tau_N,\nu_N} (\mathbf{K}^{\sigma_N}_{\tau_N,\nu_N})^\dagger \mathbf{K}^{\sigma_N}_{\tau_N,\nu_N} \right) \cdots \mathbf{K}^{\sigma_1}_{\tau_1,\nu_1} \\ &= \sum_{\tau_1,\ldots,\tau_{N-1}} \sum_{\nu_1,\ldots,\nu_{N-1}} (\mathbf{K}^{\sigma_1}_{\tau_1,\nu_1})^\dagger \cdots \left(\kappa^{\sigma_N\dagger} \kappa^{\sigma_N} \right) \cdots \mathbf{K}^{\sigma_1}_{\tau_1,\nu_1}\end{aligned} \tag{3.11}$$

But $\kappa^{\sigma_N^\dagger}\kappa^{\sigma_N} = \mathbb{I}$, so we can identically repeatedly simplify this sum until we have

$$\sum_{\tau_1}\sum_{\nu_1}(\mathbf{K}^{\sigma_1}_{\tau_1,\nu_1})^\dagger \mathbf{K}^{\sigma_1}_{\tau_1,\nu_1} = 1$$

showing that diagonal blocks trace to 1. Taking the trace of the off-diagonal blocks proceeds similarly; however, since $\sigma_j \neq \sigma'_j$ for some j, we will eventually have the term $\kappa^{\sigma'_j\dagger}\kappa^{\sigma_j} = 0$, which will send the trace of the off-diagonal block to zero. □

While Lemma 1 gives a simple constraint that can easily parameterize TP LPDOs, it is too strict. Using a maximum likelihood approach similar to Torlai et al. [2020] but *constraining the LPDO cores to lie on a Stiefel manifold*[11], we find that the procedure cannot learn even elementary 2-qubit gates (we compare the true Choi matrix of a 2-qubit CNOT and the learned Choi matrix in Figure 3.12). This assumption forces each core to *independently* contribute to the TP requirement, preventing a TP violation in one core to be 'compensated' by another core. Thus, such a restriction to independent constraints on each core does not produce a sufficiently general parameterization of Choi matrices.

3.3.3 A Necessary and Sufficient Constraint for TP LPDOs

Here, we propose a *more general* pair of conditions on the tensor cores so the LPDO satisfies a global TP constraint. As we will see, while these constraints are necessary and sufficient for the LPDO to be TP, translating these constraints into a parameterization faces serious difficulties.

[11] We compute the gradient of the log-likelihood with respect to each tensor core and apply the algorithm from Wen and Yin [2013b] to retract the gradient at each core onto the local Stiefel manifold.

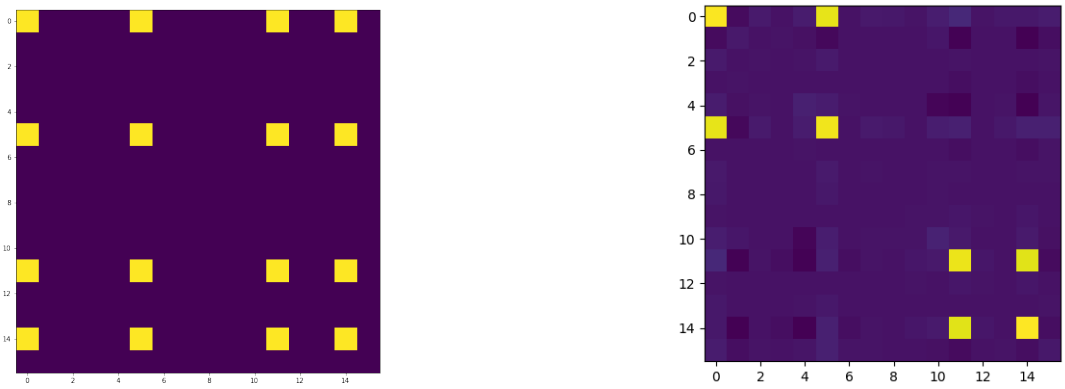


Figure 3.12: **Left:** Choi matrix of 2-qubit CNOT gate. The yellow and dark-blue elements correspond to non-zero and zero entries of the matrix. **Right:** Choi matrix of TP LPDO learned from tomography data whose cores are constrained according to Lemma 1. The learned matrix fails to capture the non-zero elements in the off-diagonal blocks.

We begin by considering the 'bottom half' of the LPDO, consisting of cores $\mathcal{K}_1, \ldots, \mathcal{K}_N$ without their conjugates. If we fuse indices ν_j, τ_j into a single index ω_j so each boundary (non-boundary) core is a third (fourth) order tensor, we can view the resulting tensor network $\mathcal{M}$ as a matrix product operator (MPO) [Murg et al., 2008] with cores $\mathbf{M}_j$. If we treat $\mathcal{M}$ as the tensor network decomposition of a matrix $[\mathbf{M}]^{\boldsymbol{\sigma}}_{\boldsymbol{\omega}}$ with rows associated with $\boldsymbol{\omega}$ and columns associated with $\boldsymbol{\sigma}$, then as in Remark 1, the TP constraint can be stated as the condition $\mathbf{M}^\dagger\mathbf{M} = \mathbb{I}$. Thus, we can also think of the problem of finding TP LPDOs as one of finding an MPO decomposition $\mathcal{M}$ of an isometry $\mathbf{M}$ living on the Stiefel manifold. Now, we state an alternative characterization of the TP constraint[12], followed by a geometric interpretation.

Theorem 10 (Necessary and Sufficient Condition for TP LPDO)**.** *Let* $[\mathbf{M}_j]^{\sigma_j}_{\omega_j,\mu_j,\mu_{j-1}}$ *be an MPO core associated with a reshaped LPDO core (as in Figure 3.13)* $[\mathbf{K}_j]^{\sigma_j}_{\tau_j,\nu_j,\mu_j,\mu_{j-1}}$ *where* τ_j, ν_j *are fused into a single index* ω_j*. Then, the LPDO is TP if and only if the MPO cores* $\mathbf{M}_j$ *satisfy the conditions in Figure 3.14a and 3.14b.*

[12] A very similar pair of constraints were given in Cirac et al. [2017] and Şahinoğlu et al. [2018] for MPOs representing unitary matrices, but under the assumption that all cores $\mathbf{M}_j$ are identical and the MPO has periodic boundary conditions. In that setting, these constraints only parameterize a subset of "simple" unitary MPOs, and require slightly stronger assumptions (and different proof techniques) to prove necessity.

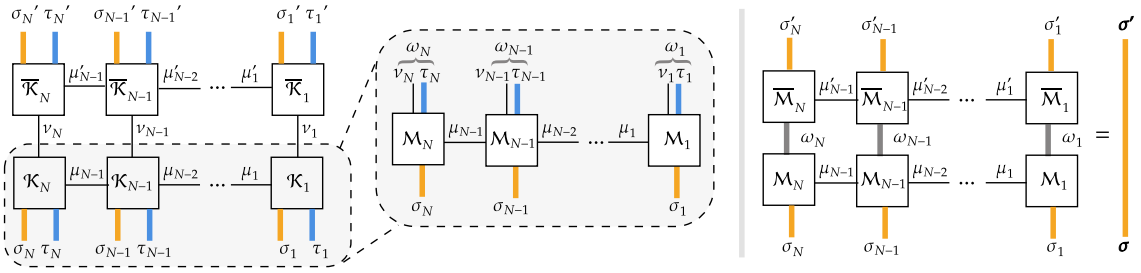
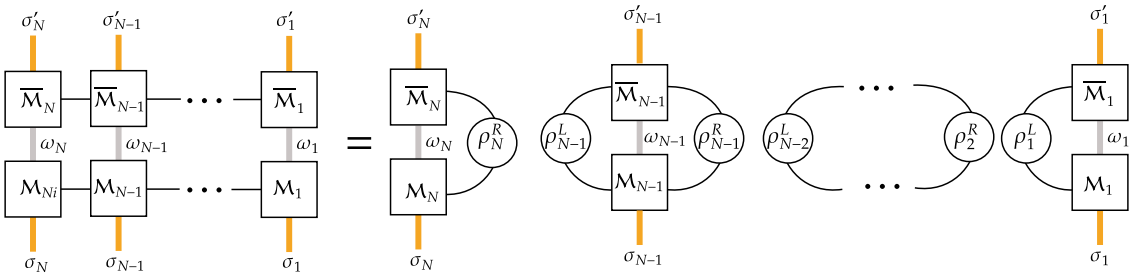
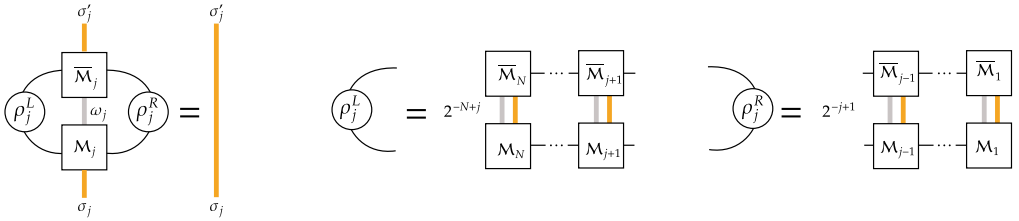


Figure 3.13: **Left:** Viewing the 'bottom-half' of an LPDO as an MPO by fusing the indices $\{\nu_i, \tau_i\}$ into a single index ω_i, **Right:** TP constraint specified on MPO



(a) The MPO must be decomposable into factors with bonds of unit Schmidt-rank



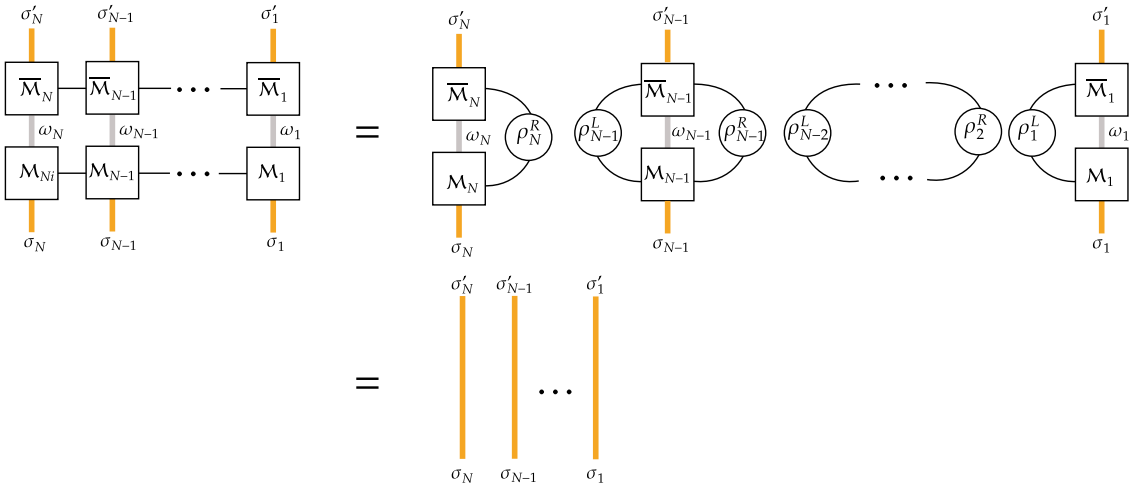
(b) Factors must be identity

(c) Definitions of ρ_j^L and ρ_j^R

Figure 3.14: (a-b) Pair of conditions which together are necessary and sufficient for MPO $\mathcal{M}$ of an LPDO to be TP. (c) Definition of the matrices ρ_j^L and ρ_j^R appearing in the above conditions, which are given by a weighted trace over all σ_j indices to the left or right of the i-th core.

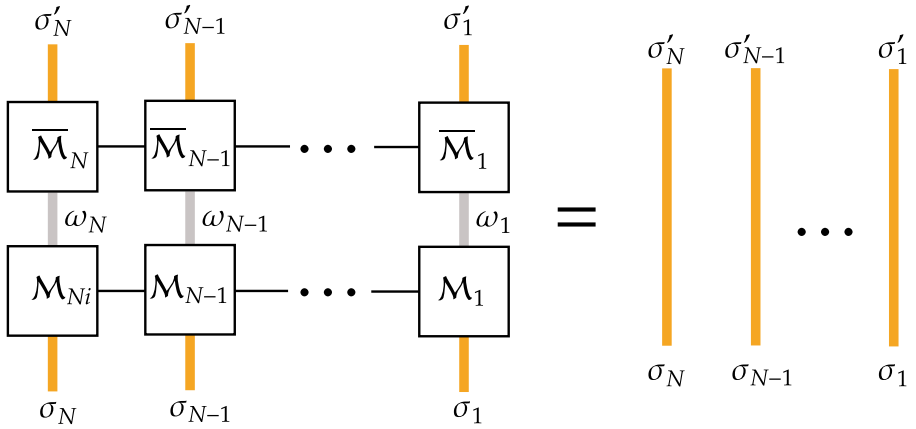
Proof.

($\Leftarrow$) Assume the conditions in Figures 3.14a and 3.14b hold. Then,

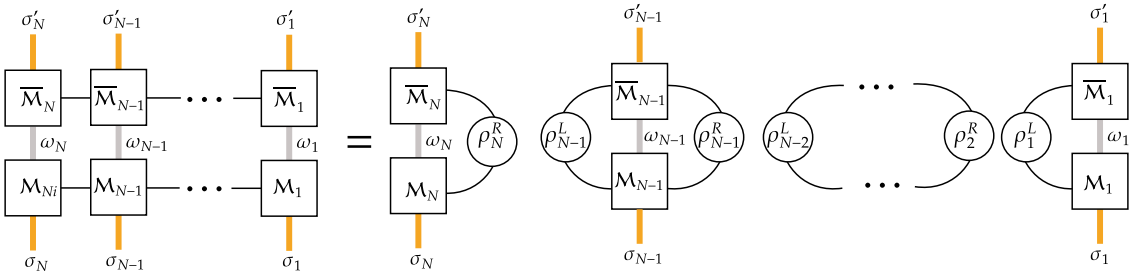


(⇒)

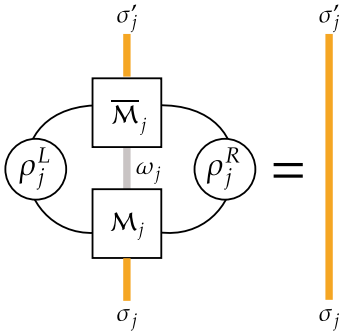
We start by assuming that that the LPDO is trace-preserving, and thus satisfies the following condition.



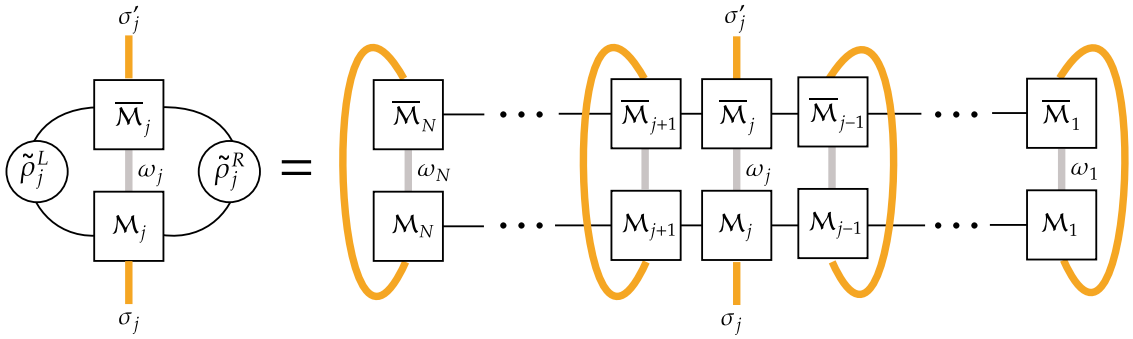
Now, from Theorem 2 in Perez-Garcia et al. [2006], we know that for every i-th core, there exist matrices ρ_j^L and ρ_j^R such that our LPDO can be written as



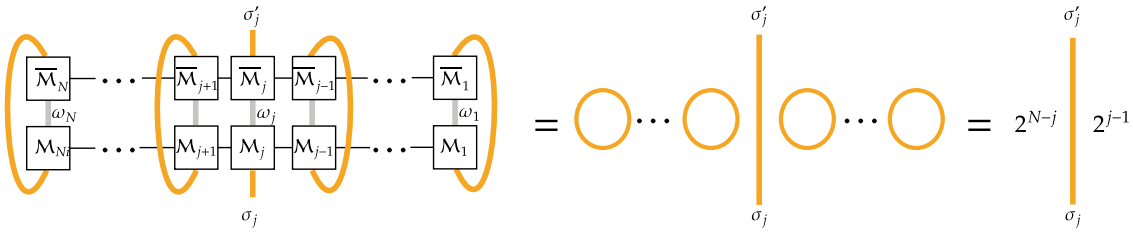
with



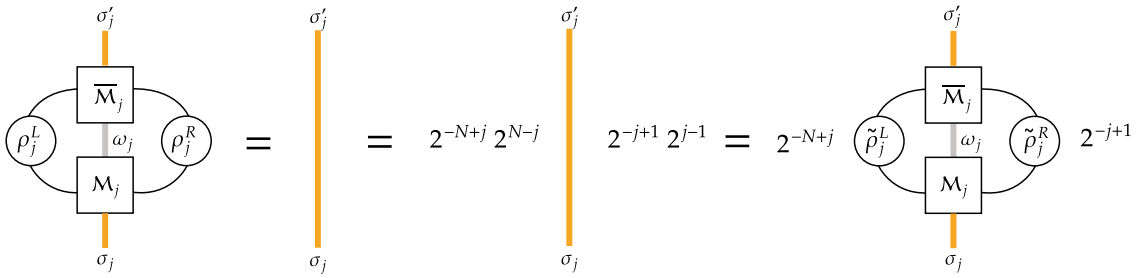
Now, to identify ρ_j^L and ρ_j^R, let us define $\tilde{\rho}_j^L$ and $\tilde{\rho}_j{}^R$ without the scalar factors 2^{-N+j} and 2^{-j+1} in Figure 3.14c, and consider the contracted LPDO representation over σ'_{-j}:



But by the TP assumption,



Thus, $\rho_j^L = 2^{-N+j}\ \tilde{\rho}_j^L$ and $\rho_j^R = 2^{-j+1}\ \tilde{\rho}_j^R$. So, we have



□

Remark 2. *Theorem 10 states that if an LPDO $\mathbf{\Lambda}_\theta$ is TP, its corresponding MPO $\mathcal{M}$ must satisfy the condition in Figure 3.14b. This condition can be written as the requirement that $\mathbf{M}_j^\dagger \mathbf{G}_j \mathbf{M}_j = \mathbb{I}$ where $\mathbf{G}_j = (\boldsymbol{\rho}_j^L \otimes \mathbb{I} \otimes \boldsymbol{\rho}_j^R)$. Thus, the TP condition on LPDOs requires that each MPO core $\mathbf{M}_j$ be an orthonormal 2-frame under the metric $\mathbf{G}_j$ induced by the other tensor cores.*

Theorem 1 illustrates the difficulty in extracting a simple parameterization of TP LPDOs; modifying any core $\mathcal{M}_j$ affects ρ_{-j}^L and ρ_{-j}^R on *all* the other cores, so it is unclear how to modify a core to preserve TP, or if it is even possible. Thus, the problem of finding a TP parameterization of LPDOs appears quite challenging, and likely insurmountable.

3.3.4 Summary

We discussed the potential of using locally purified density operators (LPDOs) for tractably modeling quantum channels. Although LPDOs satisfy the CP requirement of quantum channels by design, we showed that imposing the TP constraint presents further challenges. We showed that imposing a simple condition that each LPDO tensor core independently be on the Stiefel manifold is sufficient to guarantee TP, but is too strict to capture even simple 2-qubit gates. We also gave a necessary and sufficient description of TP LPDOs with a geometrical interpretation of the constraints on each core. However, converting this characterization of TP constraints into a parameterization useful in an optimization

scheme appears to be an insurmountable challenge. Thus, we may have to resort to TP-penalty type methods by Torlai et al. [2020] for learning LPDOs.

Chapter 4

Measurement Error Mitigation

In this chapter, we turn our attention to a common problem in quantum computing: measurement errors. We study classical post-processing approaches to measurement error mitigation, and discuss the potential of an expectation-maximization algorithm to scalably tackle this problem without producing negative probabilities.

4.1 Background

Measurement errors are a significant source of error [IBM, 2022, Rig, 2022, Wright et al., 2019, Hon, 2022] for today's programmable quantum computers (QCs). Currently, QCs typically consider quantum algorithms with a measurement at the end of the quantum circuit. However, advanced quantum operations, including quantum error correction [Nielsen and Chuang, 2010, Lid, 2013], involve measurements not only at the end of quantum computation but throughout the computational scheme. Thus, measurement errors could be a significant barrier to obtaining accurate computational results and are a clear bottleneck for scaling to larger QCs. Ideally, the QCs would be designed to reduce measurement errors, but this is a non-trivial task. Measurement error mitigation (MEM) techniques are a collection of classical post-processing techniques that aim to address the measurement errors inherent to the QC hardware.

Most MEM methods assume that the true classical probability distribution ($\vec{\theta}$) (ob-

tained after executing a circuit and measuring the results in the computational basis) is modified by a response matrix R to yield $\vec{p}$, a noisy probability distribution distorted due to measurement errors:

$$\vec{p} = R\vec{\theta}. \tag{4.1}$$

This model assumes that measurement errors are systematic, linear, and time-independent. Finding the response matrix requires us to assume further that the state preparation errors can be ignored when preparing computational basis states. These simplifying assumptions are primarily accurate for IBM Quantum Experience (IBMQE) [Maciejewski et al., 2020] superconducting devices, provided the readout calibration data is up to date.

Assuming that systematic measurement errors manifest as in Equation 4.1, measurement error mitigation schemes require two steps. The first step is quantum detector tomography (QDT) [Maciejewski et al., 2020] to acquire the response matrix R (see Figure 4.1). The entries of the response matrix are conditional probabilities; R_{ij} is the probability of measuring a bitstring b_i given that computational basis element $|b_j\rangle$ was prepared. In the second step of MEM, the classical probability distribution $\vec{p}$ acquired after measuring a quantum circuit U can be fed into an MEM scheme to produce a new probability distribution $\vec{\theta}$ which approximates the true distribution (for any future experiment). The most popular strategy for acquiring $\vec{\theta}$ is to implement:

$$\vec{\theta} = R^{-1}\vec{p}. \tag{4.2}$$

Acquiring R and implementing MEM faces two main challenges: *scaling* and *negative probabilities*. First, R is a $2^n \times 2^n$ matrix for an n-qubit system and in general requires 2^n calibration experiments. In other words, the dimensions of R are exponential in n, and QDT requires exponentially many experiments. Secondly, while the response matrix R is stochastic, its inverse is not[1]. As most MEM methods rely on applying R^{-1}, this leads to a 'negative probability' issue – the distributions obtained after measurement error

[1]This is very similar to the problem of obtaining negative probabilities from the spectral algorithm for HMMs discussed in Section 2.3.4.

mitigation are *quasiprobabilities*, i.e., the elements of this quasiprobability distribution sum to unity but can be negative. Since the introduction of MEM to the quantum computing literature, several attempts have addressed the scalability and negativity issues.

Various strategies have been proposed to address the scalability of the response matrix inversion method. Assuming k-locality for measurement crosstalk [Bravyi et al., 2021, Maciejewski et al., 2020] allows R to be represented as n/k different $2^k \times 2^k$ matrices. This assumption has been verified by Nation et al. [2021] and Maciejewski et al. [2020] and presents the first step in addressing the scalability issue. The unmitigated probability distributions are already sparse as the experiments required to construct these distributions are only performed for S shots, and in practice, S will not scale exponentially with qubit size due to time and computational constraints. Nation et al. [2021] relied on this sparsity to make MEM tractable. In particular, only the matrix elements of R that had bitstrings close to the observed ones (measured using Hamming distance) could be considered, while others were ignored. This *subspace reduction* of R suggested that a user could perform QDT after the experiment with the knowledge of the unmitigated experimental results. In other words, the order of QDT and the experiment in Figure 4.1 was flipped. Similarly, Mooney et al. [2021] suggested placing a precision threshold on R and MEM, such that any probabilities below $1/(10 \times s)$ are ignored. These approaches to scaling MEM have proven to be quite effective. In particular, Mooney et al. [2021], Nation et al. [2021], and Yang et al. [2022] have demonstrated GHZ state preparation on 27, 42 and 65-qubits respectively, where MEM plays a crucial part in each of these demonstrations; without MEM, the success metrics are significantly lower.

As for the negativity issues, numerous methods have been proposed, but no clear consensus has emerged. As all the scalable methods listed above aim to invert R and apply R^{-1}, the non-stochasticity of R^{-1} necessarily leads to quasiprobabilities. A common strategy is to use constrained least-squares optimization to find the probability distribution closest to the mitigated quasiprobability distribution, but it is unclear how to scale this approach. Alternatively, the negative terms can be canceled systematically by pro-

jecting to the nearest valid probability distribution, as detailed in Smolin et al. [2012]. However, this zeroes out all negative probabilities and can no longer be considered an unbiased estimator of the true distribution without measurement errors. Nachman et al. [2020] highlighted that error-prone detectors are standard across experimental fields, and unfolding techniques that are used in high-energy physics experiments can also be used to mitigate readout errors on quantum devices. In particular, Nachman et al. [2020] showed that iterative Bayesian unfolding (IBU) [D'Agostini, 1995] could be used to mitigate readout errors without compromising on non-negativity.

Our main contribution in this chapter is a scalable implementation of IBU. The original IBU formulation starts with the response matrix R, the empirical noisy distribution $\vec{p}$, and an initial guess $\vec{\theta}^0$. Then it iteratively applies Bayes' theorem to find a mitigated probability distribution $\vec{\theta}^k$ like so:

$$\theta_j^{k+1} = \sum_{i=1}^{2^n} p_i \cdot \frac{R_{ij}\theta_j^k}{\sum_m R_{im}\theta_m^k} \tag{4.3}$$

A well-known technique in the high-energy physics community, IBU is also known as the Richardson-Lucy deconvolution [Richardson, 1972, Lucy, 1974]. While IBU uses Bayes' theorem and starts with an initial guess sometimes referred to as a 'prior', it does not report a 'posterior' in any sense and is not a Bayesian method. It is more appropriate to call it iterative expectation maximization unfolding [Volobouev, 2015] as it converges to the maximum likelihood estimate [Shepp and Vardi, 1982] for a large number of iterations[2]. We provide a derivation of IBU as expectation maximization as relevant to our quantum computing setting.

While IBU does not return any quasi-probabilities in its basic form, it requires iterative matrix multiplications with R and was previously difficult to scale to many qubits. Our main challenge was to show that under reasonable assumptions about measurement crosstalk and precision, IBU can be implemented scalably. To demonstrate the scalability

[2]We do show in Section 4.3.1 how IBU can be straightforwardly extended to be a Baeysian method, when we have a Dirichlet prior on the guess.

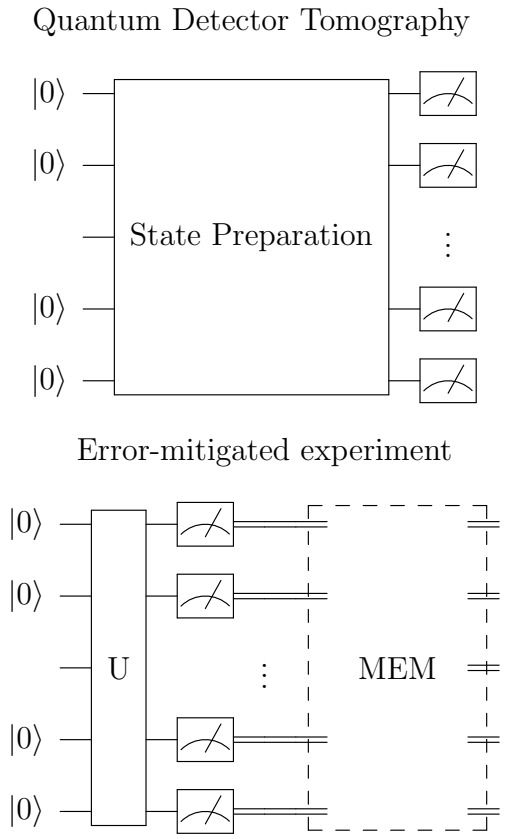


Figure 4.1: Measurement error mitigation schemes require two separate steps. (Top) Circuit for quantum detector tomography (QDT). QDT is performed to estimate the response matrix R. (Bottom) Circuit for MEM application. After applying a unitary U, the classical measurement data is fed into a MEM scheme to produce an error mitigated probability distribution.

of our method, we mitigate errors on QC data acquired by implementing the Bernstein-Vazirani algorithm [Bernstein and Vazirani, 1997] on the 27-qubit IBMQ Montreal and preparing GHZ states for the 127-qubit IBMQ Washington. The error mitigated probability distributions perform as well as or better than the current state-of-the-art M3 method [Nation et al., 2021] on our metrics *without* producing negative probabilities, albeit with slightly longer computational time.

4.2 Scalable IBU Implementation

Here, we discuss some assumptions and computational tools and tricks we use to scalably implement IBU (as in Equation 4.3) for measurement error mitigation.

Vectorized Implementation First, we show how to vectorize the IBU update rule given in Equation 4.3, enabling fast parallel computation on GPUs. With $c(z_i')$ the counts of bitstring z_i' and S shots, let $\vec{p} \in \mathbb{R}^{2^n}$ be the noisy distribution with $p_i = \frac{c(z_i')}{S}$, $\vec{\theta^k} \in \mathbb{R}^{2^n}$ be the kth guess of the error-mitigated distribution with ${\theta^k}_j = P_{\vec{\theta^k}}(Z_j)$, and $R \in \mathbb{R}^{2^n \times 2^n}$ be the response matrix where $R_{ij} = P(Z_i'|Z_j)$. Then, the IBU update rule can equivalently be written as:

$$\begin{aligned}
\vec{\theta^{k+1}} &= \sum_{i=1}^{2^n} p_i \cdot \frac{R_i \odot \vec{\theta^k}}{R_i \vec{\theta^k}} \\
&= \left(\vec{p}^T \cdot \mathrm{diag}\left(\frac{1}{R\vec{\theta^k}} \right) \cdot R \cdot \mathrm{diag}(\vec{\theta^k}) \right)^T \\
&= \vec{\theta^k} \odot \left(R^T \left(\vec{p} \oslash R\vec{\theta^k} \right) \right)
\end{aligned} \tag{4.4}$$

Here, $\odot$ is element-wise multiplication and $\oslash$ is elementwise division. This expression fully vectorizes the IBU update rule by writing it solely in terms of elementwise and matrix products and optimizes the order of operations to avoid constructing large intermediate matrices; only the additional memory for storing $\vec{\theta^{k+1}}$ is needed.

Tensor Product Structure of Noise Model Despite the efficient update scheme above, the memory requirements are still quite unfavorable. As previously discussed, the first big memory bottleneck is that the response matrix R grows exponentially in the number of qubits: it is $2^n \times 2^n$. To get around this issue, we assume that there is no measurement crosstalk and hence R has a convenient tensor product decomposition [Maciejewski et al., 2020, Bravyi et al., 2021, Nation et al., 2021], i.e., $R = R^{(1)} \otimes R^{(2)} \otimes \ldots \otimes R^{(n)}$. While we restrict ourselves to this scenario, our implementation can be extended

to cases where measurement crosstalk is restricted to k qubits.

With this tensor product structure on R, we now have $4n$ parameters to represent R, as opposed to the original 4^n parameters. This structure also allows matrix product $R\vec{x}$ to be computed very quickly using only matrix products and reshapes [Fackler, 2019] without any additional memory requirement. This is the simplest structure we can impose on R to obtain tractability. However, this can be relaxed into other tensor network decompositions of R as well (such as matrix product states (MPS)); the goal is to avoid the exponential scaling in parameters.

Subspace Reduction for Tractability While we managed the exponential scaling of R with a tensor product decomposition, the parameters of the latent true distribution $\vec{\theta} \in \mathbb{R}^{2^n}$ also scale exponentially in the number of qubits. To manage this, we can use the subspace reduction approach of [Nation et al., 2021]: instead of maintaining and updating a vector of probabilities over 2^n bitstrings, we only maintain and update an M-dimensional vector over select bitstrings. Specifically, we pre-select M-bitstrings (e.g. to be all those bitstrings within Hamming distance d from any bitstring in our dataset), and initialize $\vec{\theta}^0$ with non-zero values in the entries corresponding to those M bitstrings. All other entries are zero. Under such an initialization $\vec{\theta}^0$, the IBU update rule will guarantee that all entries that started off zero remain zero. Thus, we can ignore those entries and track only the $M \times 1$ sub-vector corresponding to non-zero entries.

Under this subspace reduction trick, we can no longer easily leverage the Kronecker product structure of R to easily compute $R\vec{x}$ for any $\vec{x}$. Instead, we must construct a reduced matrix $\tilde{R}$ whose rows and columns correspond to the M selected bitstrings. Unlike Nation et al. [2021], we are not solving a least squares problem, so matrix-free methods like GMRES will not be of help; we really do need to construct the reduced matrix $\tilde{R}$. Even if the matrix does not grow exponentially with the number of qubits, it may still be prohibitively large to store in memory.

Our computational solution to this problem is to build the solution to $R\vec{x}$ by weighting each column R_j by x_j and only keeping the running sum. We also use the JAX library's

built-in accelerated linear algebra routines and just-in-time compilation, vectorize as many operations as possible, and make use of a GPU for parallelism. Although we incur the cost of repeatedly computing $\tilde{R}$, we trade this off against memory constraints in storing R. Should memory constraints not be the dominant concern, we can simply cache $\tilde{R}$ after the first computation.

Worst-case Analysis Suppose we run an experiment with S shots on an n qubit system and run MEM tracking strings up to Hamming distance d from the measured bitstrings. In the worst case, there will be S different measurements, so subspace-reduced IBU tracks $S \cdot \left(\sum_{k=0}^{d} \binom{n}{k}\right) \sim O(Sn^d)$ bitstrings. The three main computational subroutines are 1) constructing the subspace-reduced response matrix, 2) matrix-vector multiplication, and 3) elementwise multiplication and division. First, the subspace-reduced response matrix will have $O(S^2 n^{2d})$ cells to be constructed. Each cell of the matrix requires a single call to each of the n single-qubit response matrices, so constructing the matrix takes $O(S^2 n^{2d+1})$ operations. Second, multiplying this matrix with a vector with $O(Sn^d)$ entries takes $O(S^2 n^{2d})$ operations. Third, the elementwise multiplication and division operations take $O(Sn^d)$ operations. Overall, a single IBU iteration scales as $O(S^2 n^{2d+1})$, far better than the standard exponential scaling.

Algorithm 3 Scalable IBU

Input: $\vec{p}$ (Vector of normalized counts for each bitstring), $\{R^{(i)}\}_{i=1}^{n}$ (n single-qubit response matrices), $\vec{\theta}^0$ (initial guess of MEM distribution), tol (the convergence tolerance)
Output: $\vec{\theta}^k$ (MEM distribution)

1: **while** $\|\vec{\theta}^{k+1} - \vec{\theta}^k\| <$ tol **do**
2: Compute $\vec{x}_1 = (R^{(1)} \otimes R^{(2)} \otimes \ldots \otimes R^{(n)})\vec{\theta}^k$ using [Fackler, 2019] if no subspace reduction, else by keeping a running sum of the columns R_j weighted by $\vec{\theta}_j^k$
3: Compute $\vec{x}_2 = \vec{p} \oslash \vec{x}_1$
$\vec{x}_3 = R^T \vec{x}_2$ using [Fackler, 2019] if no subspace reduction, else by keeping a running sum of the columns R_j^T weighted by $(\vec{x}_2)_j$
4: Compute $\vec{\theta}^{k+1} = \vec{\theta}^k \odot \vec{x}_3$.
5: **end while**
6: **return** $\vec{\theta}^k$

4.3 Iterative Bayesian Unfolding as Expectation-Maximization

Here we give the derivation showing that iterative Bayesian unfolding for MEM is an instantiation of the well-known *Expectation-Maximization* (EM) algorithm with multinomial distributions. Shepp and Vardi [1982] have noted connections between IBU and EM for Poisson variables. In our setting, we have at most 2^n discrete observations each with an associated probability, so our measurements come from a multinomial distribution parameterized by $\vec{\theta} \in \mathbb{R}^{2^n}$. Consequently, our derivation shows the IBU-EM connection for the multinomial distribution typical in similar quantum computing settings.

EM is a standard maximum-likelihood approach to estimating the parameters of latent variable models where optimizing the likelihood analytically and directly is not possible. The EM algorithm iterates between an *E-step* which estimates a conditional expected likelihood under a given choice of model parameters, and an *M-step* which updates the model parameters to maximize the aforementioned conditional likelihood. EM maximizes a lower bound of the likelihood at every iteration, and thus every parameter update increases the likelihood. The method is guaranteed to converge to a local maximum.

Setup Assume all measurements are made in a fixed basis. Let Z be the discrete random variable denoting n-length bitstrings with $P(Z)$ being the distribution over bitstrings prepared by the quantum computer. Due to measurement error, the distribution over bitstrings that are actually measured is different, and we denote this with random variable Z' with distribution $P(Z')$. Suppose we collect a dataset of S shots $\mathbf{Z'} = \{z'_1, \ldots, z'_S\}$ sampled IID from $P(Z')$, and that we have access to the error model $P(Z'|Z)$. Our goal is to estimate $P_\theta(Z)$, where $\vec{\theta}$ is 2^n-dimensional vector summing to 1 that parameterizes the true multinomial distribution over bitstrings prepared by our quantum computer.

Why EM To see why EM is needed, observe that the likelihood of observing our data given a choice of parameter $\vec{\theta}$ can be written as:

$$P(\mathbf{Z}';\vec{\theta}) = \prod_{i=1}^{S}\sum_{j=1}^{2^n} P(z_i'|z_j)P_\theta(z_j) \tag{4.5}$$

Although we can take the log likelihood to eliminate the product, we are left an irreducible sum inside the log that makes it difficult to analytically solve for the optimal parameters:

$$\ell(\vec{\theta};\mathbf{Z}') = \log P(\mathbf{Z}';\vec{\theta}) = \sum_{i=1}^{S}\log\left(\sum_{j=1}^{2^n} P(z_i'|z_j)P_\theta(z_j)\right) \tag{4.6}$$

EM is designed to get around this issue by instead maximizing a lower bound of the log-likelihood $\ell(\vec{\theta};\mathbf{Z}')$.

Expectation-Maximization We begin by observing that if we knew the parameters $\vec{\theta}$, we could easily compute the joint probability of any observed bitstring and the 'true' bitstring (i.e., the bitstring that would have been measured except for measurement error) as $P(z_i', z_j;\vec{\theta}) = P(z_i'|z_j)P_\theta(z_j)$. At the same time, if we knew the joint probability $P(z_i', z_j;\vec{\theta})$ of any observed bitstring and any given latent bitstring, we would be able to invert the previous equation to infer parameters $\vec{\theta}$. EM turns this logic into an iterative algorithm that alternates between computing the conditional expected joint log likelihood of observed and latent bitstrings given parameters $\vec{\theta}^k$ and finding the parameter $\vec{\theta}^{k+1}$ that maximizes this conditional expected log likelihood. By Jensen's inequality, we can manipulate Eq 4.6 as:

$$\begin{aligned}
\ell(\vec{\theta};\mathbf{Z}') &= \sum_{i=1}^{S}\log\left(\sum_{j=1}^{2^n} P(z_i'|z_j)P_\theta(z_j)\right)\\
&= \sum_{i=1}^{S}\log\left(\sum_{j=1}^{2^n} P(z_i'|z_j)P_\theta(z_j)\frac{P_{\theta^k}(z_j|z_i')}{P_{\theta^k}(z_j|z_i')}\right)\\
&= \sum_{i=1}^{S}\log\left(\sum_{j=1}^{2^n} P_{\theta^k}(z_j|z_i')\frac{P(z_i'|z_j)P_\theta(z_j)}{P_{\theta^k}(z_j|z_i')}\right)\\
&\geq \sum_{i=1}^{S}\sum_{j=1}^{2^n} P_{\theta^k}(z_j|z_i')\log\left(\frac{P(z_i', z_j;\vec{\theta})}{P_{\theta^k}(z_j|z_i')}\right)\\
&\equiv \hat{\ell}(\vec{\theta}|\vec{\theta}^k;\mathbf{Z}')
\end{aligned} \tag{4.7}$$

We can maximize this lower bound of the log-likelihood $\hat{\ell}(\vec{\theta}|\vec{\theta}^k; \mathbf{Z}')$ locally by updating the parameter at a given time-step $\vec{\theta}^k$ as follows:

$$\begin{aligned}\vec{\theta}^{k+1} &= \arg\max_{\vec{\theta}} \sum_{i=1}^{S}\sum_{j=1}^{2^n} P_{\theta^k}(z_j|z_i') \log\left(\frac{P(z_i', z_j; \vec{\theta})}{P_{\theta^k}(z_j|z_i')}\right) \\ &= \arg\max_{\vec{\theta}} \sum_{i=1}^{S}\sum_{j=1}^{2^n} P_{\theta^k}(z_j|z_i') \log P(z_i', z_j; \vec{\theta})\end{aligned} \tag{4.8}$$

Define $\ell(\vec{\theta}|\vec{\theta}^k; \mathbf{Z}') := \sum_{i=1}^{S}\sum_{j=1}^{2^n} P_{\theta^k}(z_j|z_i') \log P(z_i', z_j; \vec{\theta})$, and note that maximizing $\hat{\ell}(\vec{\theta}|\vec{\theta}^k; \mathbf{Z}')$ is equivalent to maximizing $\ell(\vec{\theta}|\vec{\theta}^k; \mathbf{Z}')$. EM consists of an E-step which involves computing $\ell(\vec{\theta}|\vec{\theta}^k; \mathbf{Z}')$ and an M-step which involves computing $\arg\max_{\vec{\theta}} \ell(\vec{\theta}|\vec{\theta}^k; \mathbf{Z}')$. For computational reasons, we will combine these into a single step for updating parameter $\vec{\theta}^k$ to $\vec{\theta}^{k+1}$. To locally maximize $\ell(\vec{\theta}|\vec{\theta}^k; \mathbf{Z}')$, we can take the derivative (using the Lagrange multiplier λ to enforce the normalization constraint that $\sum_j \vec{\theta}_j = 1$).

$$\begin{aligned}&\frac{\partial}{\partial\theta_j}\left(\sum_{i=1}^{S}\sum_{j'=1}^{2^n} P_{\theta^k}(z_{j'}|z_i') \log P(z_i', z_{j'}; \vec{\theta}) + \lambda\left(1 - \sum_{j'=1}^{2^n}\theta_{j'}\right)\right) \\ &= \frac{\partial}{\partial\theta_j}\left(\sum_{i=1}^{S}\sum_{j'=1}^{2^n} P_{\theta^k}(z_{j'}|z_i') \log P_\theta(z_{j'}) + \lambda\left(1 - \sum_{j'=1}^{2^n}\theta_{j'}\right)\right) \\ &= \frac{\partial}{\partial\theta_j}\left(\sum_{i=1}^{S}\sum_{j'=1}^{2^n} P_{\theta^k}(z_{j'}|z_i') \log \theta_{j'} + \lambda\left(1 - \sum_{j'=1}^{2^n}\theta_{j'}\right)\right) \\ &= \left(\sum_{i=1}^{S} P_{\theta^k}(z_j|z_i')\right)\frac{1}{\theta_j} - \lambda\end{aligned} \tag{4.9}$$

Setting this to zero, we get $\theta_j = \frac{1}{\lambda}\left(\sum_{i=1}^{S} P_{\theta^k}(z_j|z_i')\right)$, and by the requirement that $\sum_j \theta_j = 1 \Rightarrow \sum_j \frac{1}{\lambda}\left(\sum_{i=1}^{S} P_{\theta^k}(z_j|z_i')\right) = 1 \Rightarrow \lambda = \sum_{i=1}^{S}\sum_j P_{\theta^k}(z_j|z_i') = \sum_{i=1}^{S} 1 = S$. Substituting this back in, we get our update rule for each element θ_j of $\vec{\theta}$ that locally maximizes the log-likelihood given guess $\vec{\theta}^k$:

$$\theta_j^{k+1} = \frac{1}{S}\sum_{i=1}^{S} P_{\theta^k}(z_j|z_i') \tag{4.10}$$

Note that by Bayes' rule, we have $P_{\theta^k}(z_j|z'_i) = \frac{P(z'_i|z_j)P_{\theta^k}(z_j)}{\sum_m P(z'_i|z_m)P_{\theta^k}(z_m)}$. Additionally, instead of summing over every bitstring in the dataset, we can re-write it as a sum over all possible bitstrings using bitstring counts $c(z'_i)$. Doing so, we end up with the IBU update rule:

$$\theta_j^{k+1} = \sum_{i=1}^{2^n} \frac{c(z'_i)}{S} \cdot \frac{P(z'_i|z_j)P_{\theta^k}(z_j)}{\sum_m P(z'_i|z_m)P_{\theta^k}(z_m)} \tag{4.11}$$

Convergence of EM From Equation 4.7, we know that $\hat{\ell}(\vec{\theta}|\vec{\theta^k}; \mathbf{Z}')$ lower bounds the log-likelihood of the data $\ell(\vec{\theta}; \mathbf{Z}')$, i.e., that $\ell(\vec{\theta}; \mathbf{Z}') \geq \hat{\ell}(\vec{\theta}|\vec{\theta^k}; \mathbf{Z}')$. We can further show that these two functions coincide at the current choice of parameter values $\vec{\theta} = \vec{\theta^k}$:

$$\begin{aligned}
\hat{\ell}(\vec{\theta^k}|\vec{\theta^k}; \mathbf{Z}') &= \sum_{i=1}^{S}\sum_{j=1}^{2^n} P_{\theta^k}(z_j|z'_i) \log\left(\frac{P(z'_i, z_j; \vec{\theta^k})}{P_{\theta^k}(z_j|z'_i)}\right) \\
&= \sum_{i=1}^{S}\sum_{j=1}^{2^n} P_{\theta^k}(z_j|z'_i) \log P(z'_i; \vec{\theta^k}) \\
&= \sum_{i=1}^{S} \log P(z'_i; \vec{\theta^k}) \sum_{j=1}^{2^n} P_{\theta^k}(z_j|z'_i) \\
&= \sum_{i=1}^{S} \log P(z'_i; \vec{\theta^k}) \cdot 1 \\
&= \ell(\vec{\theta^k}; \mathbf{Z}')
\end{aligned} \tag{4.12}$$

Together, these facts imply that any choice of $\vec{\theta^{k+1}}$ which improves $\hat{\ell}(\vec{\theta^{k+1}}|\vec{\theta^k}; \mathbf{Z}')$ from the current estimate $\hat{\ell}(\vec{\theta^k}|\vec{\theta^k}; \mathbf{Z}')$ must also improve the log-likelihood, i.e., $\hat{\ell}(\vec{\theta^{k+1}}|\vec{\theta^k}; \mathbf{Z}') > \hat{\ell}(\vec{\theta^k}|\vec{\theta^k}; \mathbf{Z}') \Rightarrow \ell(\vec{\theta^{k+1}}; \mathbf{Z}') > \ell(\vec{\theta^k}; \mathbf{Z}')$ since $\ell(\vec{\theta^{k+1}}; \mathbf{Z}') \geq \hat{\ell}(\vec{\theta^{k+1}}|\vec{\theta^k}; \mathbf{Z}') > \hat{\ell}(\vec{\theta^k}|\vec{\theta^k}; \mathbf{Z}') = \ell(\vec{\theta^k}; \mathbf{Z}')$. Since EM updates $\vec{\theta^k}$ to maximize $\hat{\ell}(\vec{\theta}|\vec{\theta^k}; \mathbf{Z}')$ locally, the log-likelihood $\ell(\vec{\theta^k}; \mathbf{Z}')$ increases at every iteration. Indeed, the log-likelihood increases by at least the amount $\hat{\ell}(\vec{\theta^{k+1}}|\vec{\theta^k}; \mathbf{Z}') - \hat{\ell}(\vec{\theta^k}|\vec{\theta^k}; \mathbf{Z}')$. The algorithm iterates until it reaches a stationary point of $\hat{\ell}(\vec{\theta}|\vec{\theta^k}; \mathbf{Z}')$. Since $\hat{\ell}(\vec{\theta}|\vec{\theta^k}; \mathbf{Z}')$ is tangent to the log likelihood $\ell(\vec{\theta}; \mathbf{Z}')$ at $\vec{\theta} = \vec{\theta^k}$ [Bishop and Nasrabadi, 2006], the termination also coincides with a stationary point of the log-likelihood. Thus, EM converges to a local stationary point of the log likelihood.

4.3.1 Maximum a Posteriori IBU

Here we suggest a simple Bayesian extension of IBU: allowing us to impose a prior on the parameters $P(\vec{\theta})$ and identifying the maximum a posteriori (MAP) estimate of the parameters $P(\vec{\theta}|\mathbf{Z}')$. We begin by observing that:

$$\log P(\vec{\theta}|\mathbf{Z}') = \log P(\mathbf{Z}'|\vec{\theta}) + \log P(\vec{\theta}) - \log P(\mathbf{Z}') \tag{4.13}$$

Since we wish to find the choice of $\vec{\theta}$ at which the posterior is maximized, we ignore $\log P(\mathbf{Z}')$. Further, by Equation 4.7, we know that $\log P(\mathbf{Z}'|\vec{\theta}) \geq \hat{\ell}(\vec{\theta}|\vec{\theta}^k; \mathbf{Z}')$. Similar to the previous approach, instead of directly maximizing the posterior, we seek to maximize the lower-bound:

$$\vec{\theta}^{k+1} = \arg\max_{\vec{\theta}} \hat{\ell}(\vec{\theta}|\vec{\theta}^k; \mathbf{Z}') + \log P(\vec{\theta}) \tag{4.14}$$

For our purposes, we assume that the prior over parameters is specified as a Dirichlet distribution with parameter $\boldsymbol{\alpha} \in \mathbb{R}^{2^n}$, and let $K = \sum_j \alpha_j$. This is a natural choice as the Dirichlet distribution is the conjugate prior for multinomial distribution. Thus, $P(\vec{\theta}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{j=1}^{2^n} \theta_j^{\alpha_j - 1}$ where $B(\cdot)$ is the beta function. Taking the derivative:

$$\begin{aligned}
&\frac{\partial}{\partial \theta_j}\left(\hat{\ell}(\vec{\theta}|\vec{\theta}^k; \mathbf{Z}') + \log P(\vec{\theta}) + \lambda\left(1 - \sum_{j'=1}^{2^n} \theta_{j'}\right)\right) \\
&= \left(\sum_{i=1}^{S} P_{\theta^k}(z_j|z_i')\right)\frac{1}{\theta_j} - \lambda + \frac{\partial}{\partial \theta_j}\left(\log P(\vec{\theta})\right) \\
&= \left(\sum_{i=1}^{S} P_{\theta^k}(z_j|z_i')\right)\frac{1}{\theta_j} - \lambda + \frac{\partial}{\partial \theta_j}\log\left(\frac{1}{B(\boldsymbol{\alpha})}\prod_{j'=1}^{2^n}\theta_{j'}^{\alpha_{j'}-1}\right) \\
&= \left(\sum_{i=1}^{S} P_{\theta^k}(z_j|z_i')\right)\frac{1}{\theta_j} - \lambda + \frac{\partial}{\partial \theta_j}\sum_{j'=1}^{2^n}(\alpha_{j'}-1)\log\left(\theta_{j'}\right) \\
&= \left(\sum_{i=1}^{S} P_{\theta^k}(z_j|z_i')\right)\frac{1}{\theta_j} - \lambda + \frac{\alpha_j - 1}{\theta_j}
\end{aligned} \tag{4.15}$$

Setting this to zero, we get $\theta_j = \frac{1}{\lambda}\left((\alpha_j - 1) + \sum_{i=1}^{S} P_{\theta^k}(z_j|z_i')\right)$, and by the requirement that:

$$\begin{aligned}
&\sum_{j=1}^{2^n} \theta_j = 1 \\
&\Rightarrow \sum_{j=1}^{2^n} \frac{1}{\lambda}\left((\alpha_j - 1) + \sum_{i=1}^{S} P_{\theta^k}(z_j|z_i')\right) = 1 \\
&\Rightarrow \lambda = \sum_{j=1}^{2^n}(\alpha_j - 1) + \sum_{i=1}^{S}\sum_{j} P_{\theta^k}(z_j|z_i') \\
&\Rightarrow \lambda = S + K - 2^n
\end{aligned} \tag{4.16}$$

Substituting this back in, we get our update rule for each element θ_j of $\vec{\theta}$ that locally maximizes the log-likelihood given an initial guess $\vec{\theta^k}$:

$$\theta_j^{k+1} = \frac{1}{S + K - 2^n}\left((\alpha_j - 1) + \sum_{i=1}^{S} P_{\theta^k}(z_j|z_i')\right) \tag{4.17}$$

As before, we can rewrite the sum with counts $c(z_i')$ and use Bayes' rule to arrive at the update rule:

$$\theta_j^{k+1} = \frac{\alpha_j - 1}{S + K - 2^n} + \sum_{i=1}^{2^n} \frac{c(z_i')}{S + K - 2^n} \cdot \frac{P(z_i'|z_j)P_{\theta^k}(z_j)}{\sum_m P(z_i'|z_m)P_{\theta^k}(z_m)} \tag{4.18}$$

If we impose a uniform prior by setting $\alpha_j = 1$ for all j, then Equation 4.18 clearly reduces to standard IBU as in Equation 4.11. When using non-uniform priors, MAP-IBU prevents the parameters from updating too far away from the specified prior when there is limited data.

4.4 Demonstrations

We demonstrate our method on two datasets – measurements after the Bernstein-Vazirani (BV) algorithm (26 qubits) and measurements of GHZ state (up to 127 qubits). We

compare our method with M3 and focus on three metrics: accuracy (measured as success probabilities for BV and ℓ_1-error for GHZ), runtime on our machine, and total negative probabilities in the mitigated distribution. Our algorithm is implemented using JAX 0.3.10 [Bradbury et al., 2018] and run on a NVIDIA GeForce RTX 3090 GPU with 24576 MB of available memory.[3]

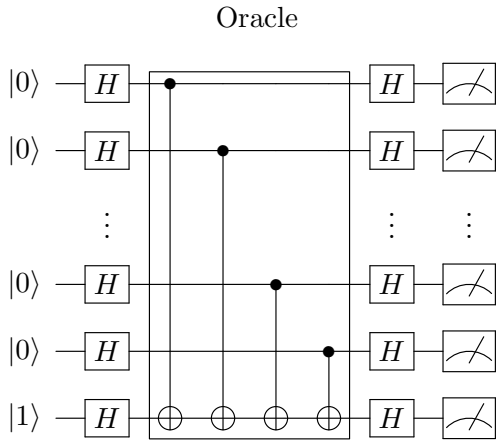


Figure 4.2: Circuit diagram for the Bernstein-Vazirani algorithm with the hidden bitstring $b = 1 \ldots 1$ [Bernstein and Vazirani, 1997, Nielsen and Chuang, 2010].

4.4.1 Bernstein-Vazirani algorithm

Problem Setting This is an oracular algorithm where a hidden bitstring $b \in \{0,1\}^n$ is known to the oracle, but when enquired with a known bitstring $x \in \{0,1\}^n$ the oracle reveals $x \oplus b$, where $\oplus$ is bit-wise addition. The goal is to guess the hidden bitstring b by making queries to the oracle. Classically, this algorithm requires $O(n)$ queries, but the BV algorithm solves this problem with $O(1)$ query. Theoretically, regardless of the problem size or the hidden bitstring, the success probability for BV is always 1 (see Figure 4.2. This expectation is violated for a real quantum computer which is subject to decoherence. The data (originally collected by Pokharel and Lidar [2022]) was taken from a publicly available repository. This work compared how the average success probability changes as a function of problem size and also claimed the quantum strategy could outperform the

[3]Our code is accessible at: `https://github.com/sidsrinivasan/PyIBU`.

classical strategy despite a decay in the success probability with increasing problem size.

Experimental Details The experiment at each problem size had 32000 shots, and we use $1/32000$ as the convergence tolerance for IBU. The primary metric is the success probability, i.e., the probability that a single guess yields the right answer. This problem employs up to 26 qubits on a bonafide quantum algorithm, and we explored whether MEM can improve this success probability.

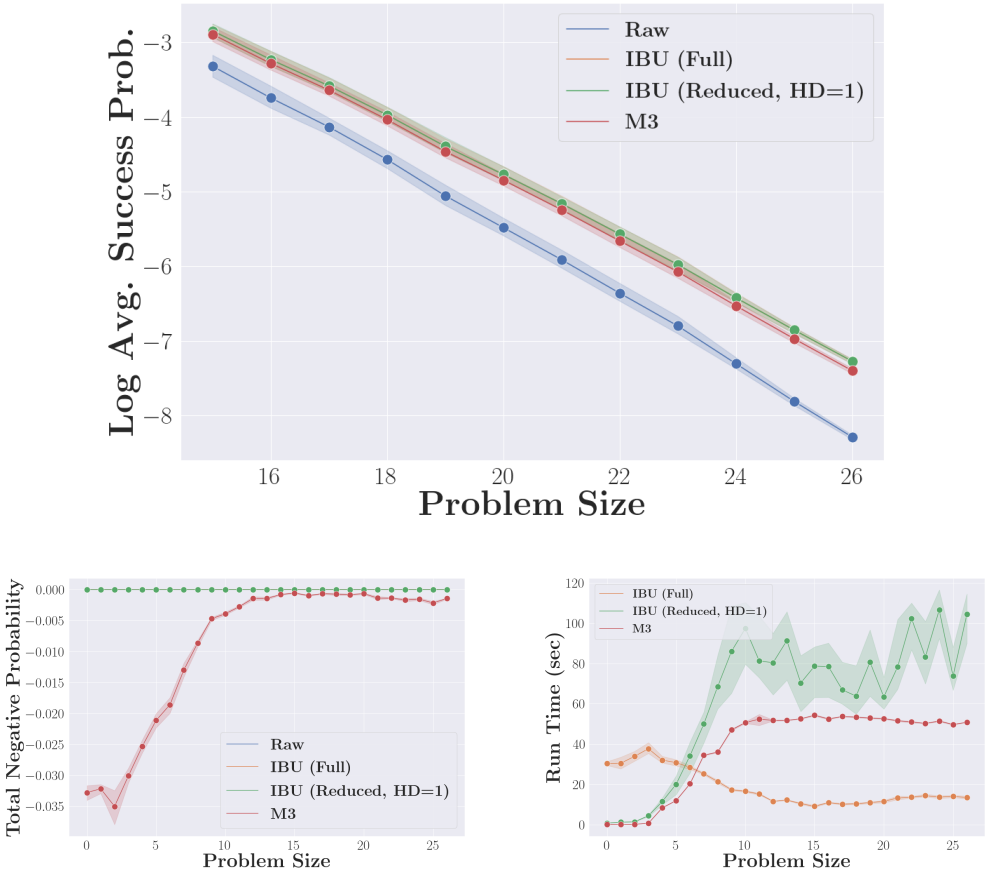


Figure 4.3: Bernstein-Vazirani Algorithm: BV algorithm was implemented on the 27-qubit IBMQ Cairo, and we compare IBU, M3, and the raw (no-MEM) result. The top figure shows log average success probability and is restricted to larger problem sizes. Full IBU and subspace-reduced IBU have identical performance. Bottom left figure shows total negative probability in the error mitigated distribution, and bottom right figure shows the run-time on our machine.

Results Our results are shown in Figure 4.3, where we zoom in on the success probability of larger sized problems. We see that a significant improvement in the success probability is possible due to measurement error mitigation. We also show the log average success probability for all problem sizes from 1 to 26, as well as the average success probability (not on the log scale) for problem sizes ranging from 1 to 14 in Figure 4.4 to provide a more complete picture of our results on this experiment. Furthermore, our IBU implementation performs about equal to or marginally better than the state-of-the-art M3 method. At the same time, the distribution produced by IBU does not contain negative probabilities, while negative probabilities are present in the quasi-probability solutions provided by M3. In terms of runtime, as the problem size increases, full IBU (without any subspace reduction) is noticeably faster than M3 while the subspace reduced version (tracking bitstrings up to Hamming distance 1) is slightly slower. Recall that this is because full IBU uses more memory while reduced IBU is less memory intensive but can be somewhat slower. For a smaller number of qubits, relying on full IBU is reasonable, but the memory constraints become the bottleneck for larger problem sizes.

Finally, we clarify that unlike the demonstration of BV [Pokharel and Lidar, 2022], we are not making any quantum speedup claims. Any classical MEM step results in a hybrid quantum-classical method, and establishing claims of quantum speedup requires greater nuance, as discussed in Pokharel and Lidar [2022] and Rønnow et al. [2014].

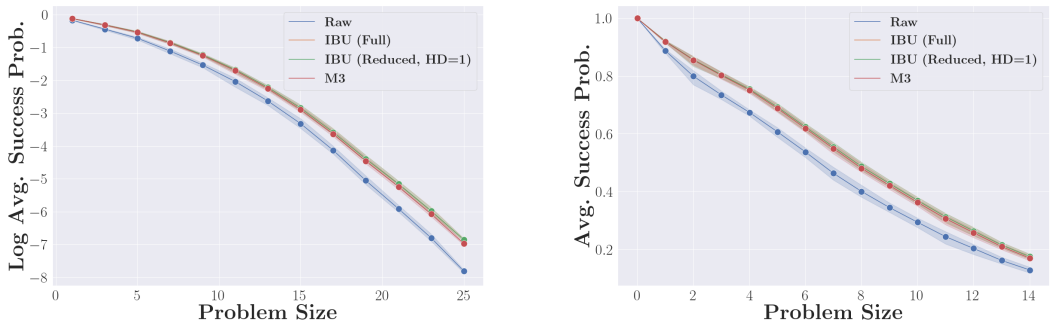


Figure 4.4: Bernstein-Vazirani: The left figure shows the log average success probability for the full range of problem sizes from 1 to 26. The right figure shows the average success probability for small problem sizes. The performance of IBU and M3 are nearly identical.

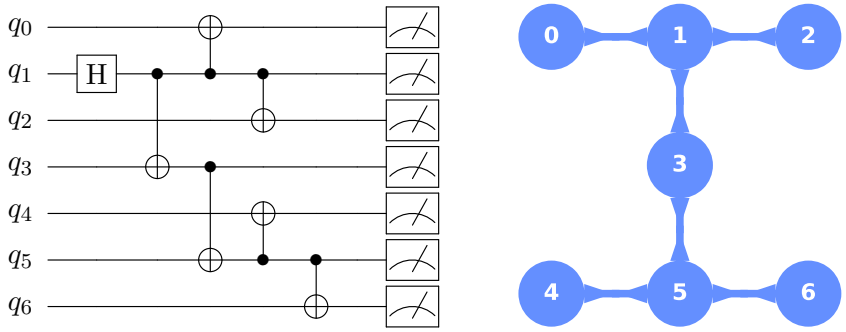


Figure 4.5: Example of GHZ generation circuit: Given a 7-qubit architecture (right), the GHZ generation circuit is created by first identifying an vertex with maximum degree (q_1) and then using breadth-first expansion and performing CNOTs accordingly to create a fully-entangled state. For our experiments, we generated GHZ on an 127-qubit device, but here we demonstrate the scheme on a 7-qubit device for simplicity.

4.4.2 Generating GHZ states

Data Generation In this experiment, we prepared the GHZ state, $|\text{GHZ}(n)\rangle = \frac{|0\rangle^{\otimes n}+|1\rangle^{\otimes n}}{\sqrt{2}}$ on the 127-qubit ibmq_washington (see Appendix A.2 for device architecture). To reduce circuit-depth, the entangling operation is implemented by parallelizing the two-qubit operations as much as is allowed by the device architecture. GHZ states are prepared by entangling n qubits. This means that as long as there is a single connection allowing for a multi-qubit entangling operation like CNOT, it is possible to create a GHZ state linearly increasing in circuit depth. Decoherence is introduced both by multi-qubit operations (which are noisy), and by the total circuit duration. In order to restrict circuit duration, we apply these CNOTs in parallelized manner. Starting from the most connected qubit (the qubit with the greatest number of connections/edges to neighouring qubits) in a superposition state $|+\rangle$, we use breadth-first search to entangle every additional qubit with CNOT gates. In effect, this amounts to prioritizing the distance 1 neighbours to entangle, followed by the distance 2 neighbours, and so on. This strategy limits circuit depth and minimizes decoherence. We give an example of this strategy in Figure 4.5.

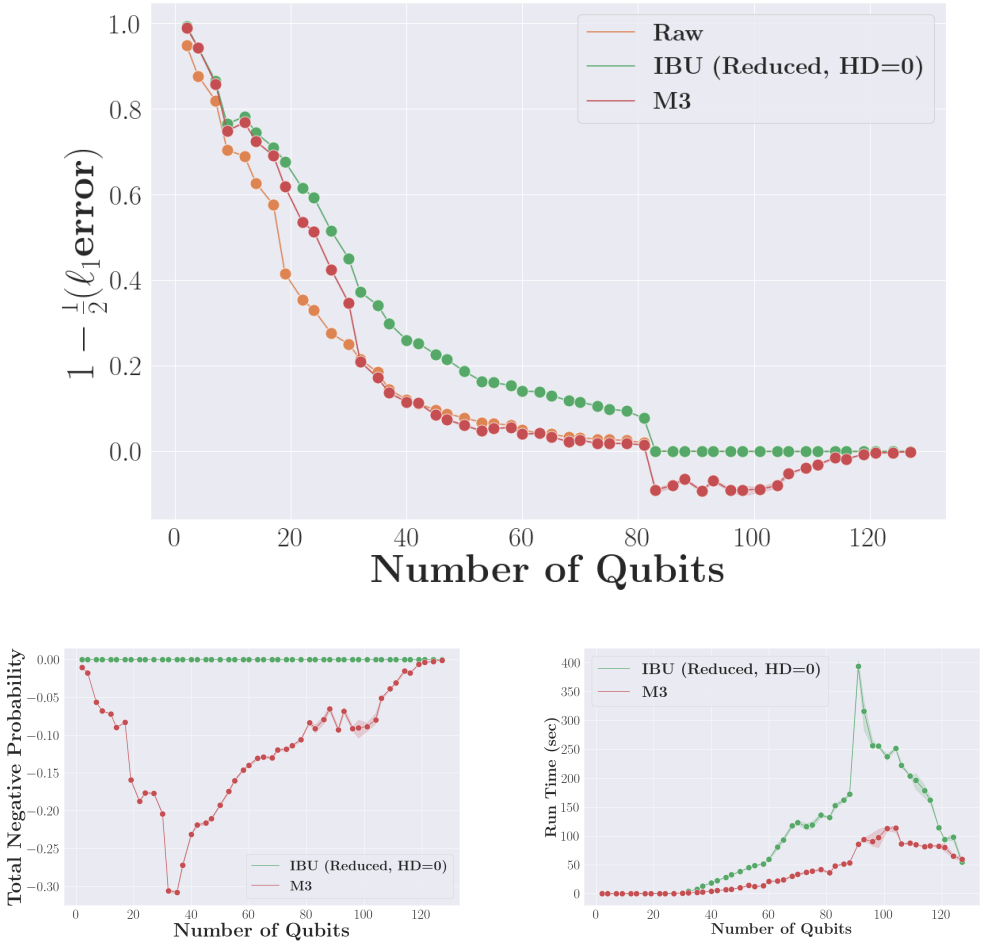


Figure 4.6: GHZ state preparation on 127-qubit IBMQ Washington: We visualize results for the full range of GHZ states we attempted to create, up to 127 qubits. Circuit errors dominate beyond $n = 81$ and MEM is not of much help. The left figure shows the normalized ℓ_1 score. At all problem sizes, IBU returns a higher value from the performance metric $1 - \frac{1}{2}(\ell_1$ error). M3 achieves negative score due to negative probabilities. Top right figure shows total negative probability in the error mitigated distribution; here M3 yields significant negative probabilities. The Bottom right figure shows run time on our machine. IBU takes longer to run compared to M3, but is reasonable in absolute terms.

Experimental Details We generated GHZ states up to $n = 127$. The experiments here were repeated for 100,000 shots and the results were bootstrapped using the observed counts. We use a convergence tolerance of $1/10000$. The primary error metric is a *normalized* ℓ_1 *score*; the ℓ_1 error between two probability distributions $\vec{p}, \vec{q}$ is $\sum_i |p_i - q_i|_1$ and lies in $[0, 2]$, and we normalize it as $1 - \frac{1}{2}(\ell_1$ error) to lie in the range $[0, 1]$ where a score of 1 implies zero ℓ_1 error. We consider the ℓ_1 error $= \sum_i |\theta_i - q_i|_1$ between the error mitigated

distribution $\vec{\theta}$ and the theoretical distribution $\vec{q}$ of bitstrings for a GHZ state measured in the Z basis (i.e., $\vec{q} = [0.5, 0, 0, \ldots, 0, 0.5]$). Our goal here was to test the capacity of our IBU implementation to scale for large systems. Owing to the memory costs, we are only able to run IBU with subspace reduction. Furthermore, we use a Hamming distance of zero, which is to say MEM modifies the probability mass only on the measured bitstrings.

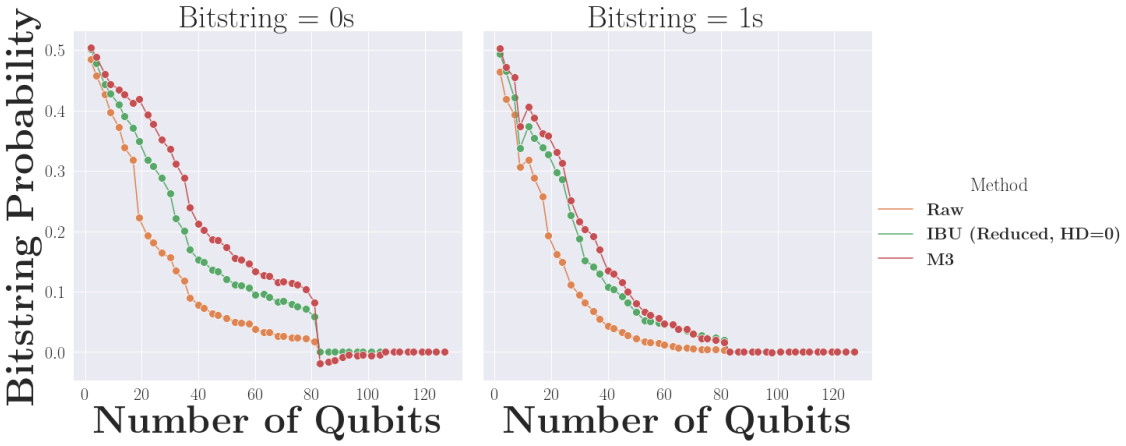


Figure 4.7: GHZ state preparation on 127-qubit IBMQ Washington: The probabilities assigned to the 'correct' bitstrings ('0^n' (left) and '1^n' (right)) with IBU, M3, and no MEM. Although IBU achieved higher normalized ℓ_1 score (Figure 4.6, M3 places greater probability on the 'correct' bitstrings and compensates for this with negative probabilities on 'wrong' bitstrings.

Results Figure 4.6 shows our results on this experiment, and we make several comments. First, we see that IBU has significantly higher normalized ℓ_1 score compared to the raw data or M3. Beyond $n = 81$, the true solution bitstrings (0^n and 1^n) were no longer the modal bitstrings due to circuit errors, so MEM (M3 and IBU) is not of much help. However, the normalized ℓ_1 score for IBU goes to zero, but the score for M3 is negative due to negative probabilities. We found that the negative probabilities produced by M3 can be quite substantial, up to $\sim 30\%$ of the probability mass. In Figure 4.7, we visualize the probabilities assigned to the 'correct' bitstrings ('0^n' and '1^n') with IBU, M3, and no MEM. Although IBU achieved higher normalized ℓ_1 score (Figure 4.6), M3 places greater probability on the 'correct' bitstrings and compensates for this with negative probabilities on 'wrong' bitstrings. While this may appear advantageous, we emphasize that it simply

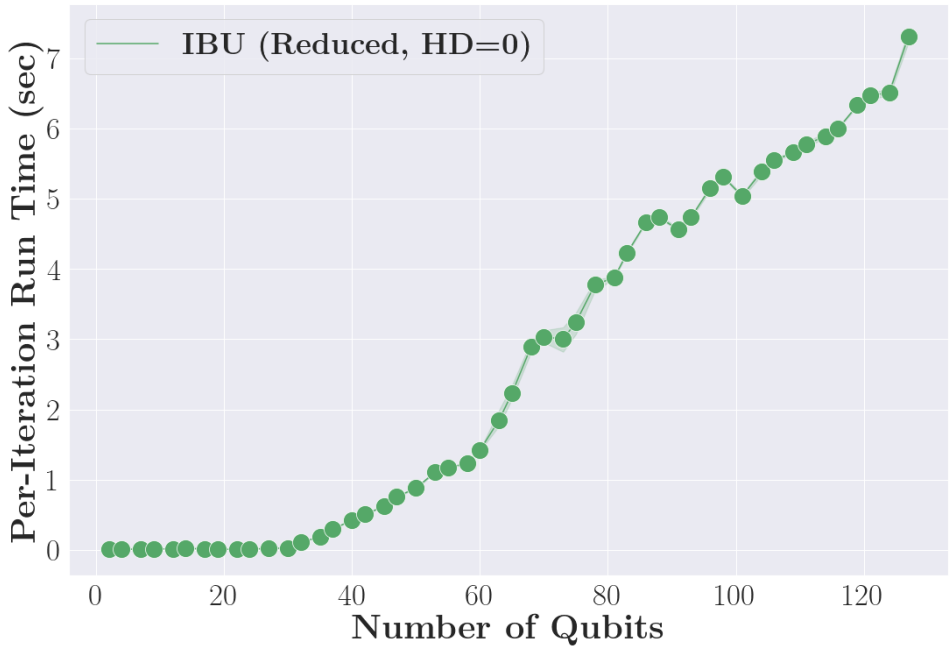


Figure 4.8: GHZ state preparation on 127-qubit IBMQ Washington (Per-Iteration Run-time): We give the time taken by a single IBU iteration (to disregard the effect of varying number of iterations till convergence) under Hamming distance 0. The run-time scales linearly with the number of qubits (when using Hamming distance 0) and the number of shots, matching theoretical expectations and showing that IBU is scalable to higher number of qubits.

happens to yield good results on this experiment. In reality, the negative probabilities show that M3's error mitigated distribution deviates from the true solution (which assigns zero probability to these bitstrings), so the normalized ℓ_1 score gives a more reliable measure of whether the error mitigated distribution matches the true distribution. Indeed, Figure 4.7 we see that M3 assigns negative probabilities to the 'correct' bitstring ('0^n') at $n > 81$. Finally, we note that the cost of achieving better ℓ_1 error is longer run-time: IBU is noticeably slower than M3. Nevertheless, in absolute terms IBU is able to perform MEM on 81 qubits in a little over 2 minutes on our machine. IBU run-time declines past $n = 81$ as iterations do not much modify the guess and there is not much room for progress. One can always choose a higher tolerance to reduce the number of IBU iterations and hence

run-time. Finally, in Figure 4.8, we show the per-iteration runtime of IBU. This disregards the effect that the convergence tolerance might have on the run-time. As expected, the graph is essentially linear in the number of qubits.

4.5 Summary

We showed that IBU, an expectation-maximization algorithm, can be scalably implemented by vectorizing operations and using GPU parallelism. We demonstrated our algorithm's scalability on the BV dataset with up to 26 qubits, and on GHZ states of up to 127 qubits (although results were not meaningful beyond 81 qubits). We found that IBU achieves lower errors than the state-of-the-art alternative M3 without producing negative probabilities. On the other hand, IBU takes longer than M3, although in absolute terms the run-time is still reasonable.

Chapter 5

Conclusion

In this work, we studied select ideas in 'quantum-inspired machine learning', by which we mean both the application of classical machine learning inspired by quantum information, as well as the application of classical machine learning to problems in quantum information. In studying linear models for stochastic processes, we developed and formulated HQMMs, a quantum-inspired analogue of HMMs, and showed that they are the most expressive known model class unaffected by the 'negative probability problem'. We explored connections between various models for linear stochastic processes and quantum tensor networks, showing their equivalence in the 'non-terminating limit' of the latter. We also explored the feasibility of developing a tensor network model for quantum channels that were constrained to satisfy the necessary physical constraints, but ran into apparently insurmountable challenges. Lastly, we implemented a scalable measurement error mitigation procedure for quantum computing using expectation-maximization, a classical machine learning algorithm.

We have thus looked at various classical and quantum problems that have benefited from a classical treatment, including modeling sequential data, quantum systems as tensor networks, quantum process tomography, and measurement error mitigation for quantum computing. Several common themes arose in these problems. First, we saw that there are close connections between linear stochastic processes that model sequences and quantum

tensor networks. Second, the issue of 'negative probabilities' arising from attempting to invert a stochastic matrix as part of a regression appeared in the spectral algorithm for PSRs, the estimation of quantum mixture covariance operators in kernel HQMMs, and in measurement error mitigation; we typically turned to maximum-likelihood based methods to tackle this. This is going to be a common issue when manipulating results from quantum systems, as a natural consequence of the probabilistic foundations of quantum information. Third, we saw from our discussion of quantum process tomography and measurement error mitigation that the exponential scaling of quantum systems is a common bottleneck; tensor-network methods will be immensely helpful in tractably modeling such systems.

Finally, clear directions for further work emerged in our presentation of these results. The problem of whether there are any finite-dimensional PSRs that have no equivalent finite-dimensional HQMMs (i.e., the completely positive realization problem from Section 2.5.5) remains open, as is the question of the exact correspondence between the Nadaraya-Watson-like 'quantum' Bayes rule and the kernel sum rule. Alternative approaches to identifying a trace-preserving tensor network representation of quantum channels will be of great use in applications (such as QPT) that require a physically valid representation of quantum channels. Further advances in computing hardware and computational tricks could help reduce the absolute run-time of iterative Bayesian unfolding for measurement error mitigation, and make it more competitive with matrix-inversion methods like M3. With further advances in quantum information and machine learning, we expect that the potential for cross-fertilization of ideas between these two fields that share deep mathematical foundations will only expand.

Bibliography

Quantum Error Correction. Cambridge University Press, Cambridge, UK, 2013. URL http://www.cambridge.org/9780521897877.

Honeywell Quantum, 2022. URL https://www.honeywell.com/us/en/company/quantum.

IBM Quantum, 2022. URL https://quantum-computing.ibm.com/.

Rigetti Computing, 2022. URL https://www.rigetti.com/.

P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, NJ, USA, 2007. ISBN 0691132984, 9780691132983.

S. Adhikary, S. Srinivasan, and B. Boots. Learning quantum graphical models using constrained gradient descent on the stiefel manifold. *arXiv preprint arXiv:1903.03730*, 2019.

S. Adhikary, S. Srinivasan, G. Gordon, and B. Boots. Expressiveness and learning of hidden quantum markov models. In *International Conference on Artificial Intelligence and Statistics*, pages 4151–4161. PMLR, 2020.

A.Hjørungnes and D.Gesbert. Complex-Valued Matrix Differentiation: Techniques and Key Results. 55(6):2740–2746, 2007. doi: 10.1109/TSP.2007.893762.

A. Anandkumar, D. Hsu, and S. M. Kakade. A method of moments for mixture models and hidden markov models. In *Conference on Learning Theory*, pages 33–1. JMLR Workshop and Conference Proceedings, 2012.

R. Bailly. Quadratic weighted automata: Spectral algorithm and likelihood maximization. In *Asian Conference on Machine Learning*, pages 147–163, 2011.

R. Bailly, F. Denis, and L. Ralaivola. Grammatical inference as a principal component analysis problem. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 33–40, 2009.

B. Balle, X. Carreras, F. M. Luque, and A. Quattoni. Spectral learning of weighted automata. *Machine learning*, 96(1-2):33–63, 2014a.

B. Balle, W. Hamilton, and J. Pineau. Methods of moments for learning stochastic languages: Unified presentation and empirical comparison. In *International Conference on Machine Learning*, pages 1386–1394, 2014b.

L. Benvenuti and L. Farina. A tutorial on the positive realization problem. *IEEE Transactions on automatic control*, 49(5):651–664, 2004.

E. Bernstein and U. Vazirani. Quantum Complexity Theory. *SIAM Journal on Computing*, 26(5):1411–1473, Oct. 1997. ISSN 0097-5397. doi: 10.1137/S0097539796300921.

J. Biamonte and V. Bergholm. Tensor networks in a nutshell. *arXiv preprint arXiv:1708.00006*, 2017.

J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

J. A. Bilmes et al. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. *International computer science institute*, 4(510):126, 1998.

C. M. Bishop and N. M. Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

B. Boots, G. Gordon, and A. Gretton. Hilbert space embeddings of predictive state representations. *arXiv preprint arXiv:1309.6819*, 2013.

J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

S. Bravyi, S. Sheldon, A. Kandala, D. C. Mckay, and J. M. Gambetta. Mitigating measurement errors in multiqubit experiments. *Physical Review A*, 103(4):042605, Apr. 2021. ISSN 2469-9926, 2469-9934. doi: 10/gjtqvd.

Y. Chen, M. Welling, and A. Smola. Super-samples from kernel herding. *arXiv preprint arXiv:1203.3472*, 2012.

M.-D. Choi. Completely positive linear maps on complex matrices. *Linear algebra and its applications*, 10(3):285–290, 1975.

J. Cirac and G. Sierra. Infinite matrix product states, conformal field theory, and the haldane-shastry model. *Physical Review B*, 81:104431, 2010.

J. I. Cirac, D. Perez-Garcia, N. Schuch, and F. Verstraete. Matrix product density operators: Renormalization fixed points and boundary theories. *Annals of Physics*, 378: 100–149, 2017.

S. B. Cohen, K. Stratos, M. Collins, D. P. Foster, and L. Ungar. Experiments with spectral learning of latent-variable pcfgs. In *Proceedings of the 2013 conference of the North American chapter of the Association for Computational Linguistics: human language technologies*, pages 148–157, 2013.

I. Cong, S. Choi, and M. D. Lukin. Quantum convolutional neural networks. *Nature Physics*, 15(12):1273–1278, 2019.

A. Critch, J. Morton, et al. Algebraic geometry of matrix product states. *SIGMA. Symmetry, Integrability and Geometry: Methods and Applications*, 10:095, 2014.

G. D. L. Cuevas, J. Cirac, N. Schuch, and D. Pérez-García. Irreducible forms of matrix product states: Theory and applications. *Journal of Mathematical Physics*, 58:121901, 2017.

G. D'Agostini. A multidimensional unfolding method based on Bayes' theorem. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 362(2):487–498, Aug. 1995. ISSN 0168-9002. doi: 10.1016/0168-9002(95)00274-X.

G. De las Cuevas, N. Schuch, D. Pérez-García, and J. I. Cirac. Purifications of multipartite states: limitations and constructive methods. *New Journal of Physics*, 15(12):123021, 2013.

F. Denis and Y. Esposito. Learning classes of probabilistic automata. In *International Conference on Computational Learning Theory*, pages 124–139. Springer, 2004.

F. Denis and Y. Esposito. On rational stochastic languages. *Fundamenta Informaticae*, 86(1, 2):41–77, 2008.

D. Dheeru and E. Karra Taniskidou. UCI machine learning repository, 2017. Available at: http://archive.ics.uci.edu/ml.

C. Downey, A. Hefny, B. Li, B. Boots, and G. J. Gordon. Predictive state recurrent neural networks. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2017.

P. Dupont, F. Denis, and Y. Esposito. Links between probabilistic automata and hidden markov models: probability distributions, learning models and induction algorithms. *Pattern recognition*, 38(9):1349–1371, 2005.

J. Eisert, M. Cramer, and M. B. Plenio. Colloquium: Area laws for the entanglement entropy. *Reviews of Modern Physics*, 82(1):277, 2010.

D. E. Evans and R. Høegh-Krohn. Spectral properties of positive maps on c*-algebras. *Preprint series: Pure mathematics http://urn. nb. no/URN: NBN: no-8076*, 1977.

P. L. Fackler. Algorithm 993: Efficient computation with kronecker products. *ACM Transactions on Mathematical Software (TOMS)*, 45(2):1–9, 2019.

M. Fannes, B. Nachtergaele, and R. F. Werner. Finitely correlated states on quantum spin chains. *Communications in mathematical physics*, 144(3):443–490, 1992.

T. Garipov, D. Podoprikhin, A. Novikov, and D. Vetrov. Ultimate tensorization: compressing convolutional and fc layers alike. *arXiv preprint arXiv:1611.03214*, 2016.

A. Gheondea. The three equivalent forms of completely positive maps on matrices. *Annals of the University of Bucharest*, 2010.

I. Glasser, R. Sweke, N. Pancotti, J. Eisert, and I. Cirac. Expressive power of tensor-network factorizations for probabilistic modeling. In *Advances in Neural Information Processing Systems*, pages 1496–1508, 2019.

S. Grünewälder, G. Lever, A. Gretton, L. Baldassarre, S. Patterson, and M. Pontil. Conditional mean embeddings as regressors. In *ICML*, 2012.

C. Guo, Z. Jie, W. Lu, and D. Poletti. Matrix product operators for sequence-to-sequence learning. *Physical Review E*, 98(4):042114, 2018.

Z.-Y. Han, J. Wang, H. Fan, L. Wang, and P. Zhang. Unsupervised generative modeling using matrix product states. *Physical Review X*, 8(3):031012, 2018.

A. Hefny, C. Downey, and G. J. Gordon. Supervised learning for dynamical system learning. In *Advances in neural information processing systems*, pages 1963–1971, 2015.

A. Heller. On stochastic processes derived from markov chains. *The Annals of Mathematical Statistics*, 36(4):1286–1291, 1965.

J. W. Helton and J. Nie. Sufficient and necessary conditions for semidefinite representability of convex hulls and sets. *SIAM Journal on Optimization*, 20(2):759–791, 2009.

D. Hsu, S. M. Kakade, and T. Zhang. A spectral algorithm for learning hidden markov models. *Journal of Computer and System Sciences*, 78(5):1460–1480, 2012.

W. Huggins, P. Patil, B. Mitchell, K. B. Whaley, and E. M. Stoudenmire. Towards quantum machine learning with tensor networks. *Quantum Science and technology*, 4 (2):024001, 2019.

H. Ito, S.-I. Amari, and K. Kobayashi. Identifiability of hidden markov information sources and their minimum degrees of freedom. *IEEE transactions on information theory*, 38 (2):324–333, 1992.

H. Jaeger. Observable operator models for discrete stochastic time series. *Neural Computation*, 12(6):1371–1398, 2000.

H. Jaeger, M. Zhao, and A. Kolling. Efficient estimation of ooms. *Advances in Neural Information Processing Systems*, 18, 2005.

A. Jamiołkowski. Linear transformations which preserve trace and positive semidefiniteness of operators. *Reports on Mathematical Physics*, 3(4):275–278, 1972.

B. Jiang and Y.-H. Dai. A framework of constraint preserving update schemes for optimization on stiefel manifold. *Math. Program.*, 153:535–575, 2013.

M. Jiang, S. Luo, and S. Fu. Channel-state duality. *Physical Review A*, 87(2):022310, 2013.

N. Johnston. QETLAB: A MATLAB toolbox for quantum entanglement, version 0.9. `http://qetlab.com`, Jan. 2016.

M. Kanagawa and K. Fukumizu. Recovering distributions from gaussian rkhs embeddings. In *Artificial Intelligence and Statistics*, pages 457–465, 2014.

M. Kliesch, D. Gross, and J. Eisert. Matrix-product operators and states: Np-hardness and undecidability. *Physical review letters*, 113(16):160503, 2014.

A. Klümper, A. Schadschneider, and J. Zittartz. Matrix product ground states for one-dimensional spin-1 quantum antiferromagnets. *EPL (Europhysics Letters)*, 24(4):293, 1993.

G. C. Knee, E. Bolduc, J. Leach, and E. M. Gauger. Quantum process tomography via completely positive and trace-preserving projection. *Physical Review A*, 98(6):062336, 2018.

R. L. Kosut, A. Shabani, and D. A. Lidar. Robust quantum error correction via convex optimization. *Physical review letters*, 100(2):020502, 2008.

K. Kraus. General state changes in quantum theory. *Annals of Physics*, 64(2):311–335, 1971.

J. M. Kübler, K. Muandet, and B. Schölkopf. Quantum mean embedding of probability distributions. *Physical Review Research*, 1(3):033159, 2019.

L. Li, K. G. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. S. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18:185:1–185:52, 2017.

I. A. Luchnikov, A. Ryzhov, S. N. Filippov, and H. Ouerdane. QGOpt: Riemannian optimization for quantum technologies. *SciPost Phys.*, 10:79, 2021. doi: 10.21468/SciPostPhys.10.3.079. URL `https://scipost.org/10.21468/SciPostPhys.10.3.079`.

L. B. Lucy. An iterative technique for the rectification of observed distributions. *The Astronomical Journal*, 79:745, June 1974. ISSN 0004-6256. doi: 10.1086/111605.

F. B. Maciejewski, Z. Zimborás, and M. Oszmaniec. Mitigation of readout noise in near-term quantum devices by classical post-processing based on detector tomography. *Quantum*, 4:257, Apr. 2020. doi: 10.22331/q-2020-04-24-257.

A. Mahajan, M. Samvelyan, L. Mao, V. Makoviychuk, A. Garg, J. Kossaifi, S. Whiteson, Y. Zhu, and A. Anandkumar. Tesseract: Tensorised actors for multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 7301–7312. PMLR, 2021.

M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

O. Mickelin and S. Karaman. Tensor ring decomposition. *ArXiv*, abs/1807.02513, 2018.

J. Miller, G. Rabusseau, and J. Terilla. Tensor networks for probabilistic sequence modeling. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*, 2021.

A. Monras and A. Winter. Quantum learning of classical stochastic processes: The completely positive realization problem. *Journal of Mathematical Physics*, 57(1):015219, 2016.

A. Monras, A. Beige, and K. Wiesner. Hidden quantum markov models and non-adaptive read-out of many-body states. *arXiv preprint arXiv:1002.2337*, 2010.

G. J. Mooney, G. A. L. White, C. D. Hill, and L. C. L. Hollenberg. Generation and verification of 27-qubit Greenberger-Horne-Zeilinger states in a superconducting quantum computer. *Journal of Physics Communications*, 5(9):095004, Sept. 2021. ISSN 2399-6528. doi: 10.1088/2399-6528/ac1df7.

K. Muandet, K. Fukumizu, B. Sriperumbudur, and B. Schölkopf. Kernel mean embedding of distributions: A review and beyond. *arXiv preprint arXiv:1605.09522*, 2016.

V. Murg, J. Cirac, B. Pirvu, and F. Verstraete. Matrix product operator representations. *New Journal of Physics*, 12:025012, 2008.

B. Nachman, M. Urbanek, W. A. de Jong, and C. W. Bauer. Unfolding Quantum Computer Readout Noise. *arXiv:1910.01969 [physics, physics:quant-ph]*, May 2020. URL `http://arxiv.org/abs/1910.01969`.

P. D. Nation, H. Kang, N. Sundaresan, and J. M. Gambetta. Scalable mitigation of measurement errors on quantum computers. *arXiv:2108.12518 [quant-ph]*, Aug. 2021. URL `http://arxiv.org/abs/2108.12518`.

M. A. Nielsen and I. Chuang. Quantum computation and quantum information, 2002.

M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.

A. Novikov, A. Rodomanov, A. Osokin, and D. Vetrov. Putting mrfs on a tensor train. In *International Conference on Machine Learning*, pages 811–819, 2014.

A. Novikov, D. Podoprikhin, A. Osokin, and D. P. Vetrov. Tensorizing neural networks. In *Advances in neural information processing systems*, pages 442–450, 2015.

R. Orús. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Annals of Physics*, 349:117–158, 2014.

I. Oseledets. Tensor-train decomposition. *SIAM J. Sci. Comput.*, 33:2295–2317, 2011.

D. Perez-Garcia, F. Verstraete, M. M. Wolf, and J. I. Cirac. Matrix product state representations. *arXiv preprint quant-ph/0608197*, 2006.

B. Pokharel and D. A. Lidar. Demonstration of algorithmic quantum speedup, July 2022. URL `http://arxiv.org/abs/2207.07647`.

N. Qian. On the momentum term in gradient descent learning algorithms. *Neural Netw.*, 12(1):145–151, Jan. 1999. ISSN 0893-6080. doi: 10.1016/S0893-6080(98)00116-6. URL `http://dx.doi.org/10.1016/S0893-6080(98)00116-6`.

L. R. Rabiner. An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16, 1986.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008.

W. H. Richardson. Bayesian-Based Iterative Method of Image Restoration*. *Journal of the Optical Society of America*, 62(1):55, Jan. 1972. ISSN 0030-3941. doi: 10.1364/JOSA.62.000055.

T. F. Rønnow, Z. Wang, J. Job, S. Boixo, S. V. Isakov, D. Wecker, J. M. Martinis, D. A. Lidar, and M. Troyer. Defining and detecting quantum speedup. *Science*, 345(6195): 420–424, July 2014. doi: 10.1126/science.1252319.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA, 1988. ISBN 0-262-01097-6. URL `http://dl.acm.org/citation.cfm?id=65669.104451`.

M. B. Şahinoğlu, S. K. Shukla, F. Bi, and X. Chen. Matrix product representation of locality preserving unitaries. *Physical Review B*, 98(24):245122, 2018.

C. Scheiderer. Spectrahedral shadows. *SIAM Journal on Applied Algebra and Geometry*, 2(1):26–44, 2018.

C. Schoen, E. Solano, F. Verstraete, J. Cirac, and M. M. Wolf. Sequential generation of entangled multiqubit states. *Physical review letters*, 95 11:110503, 2005.

B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. In *International conference on computational learning theory*, pages 416–426. Springer, 2001.

M. Schuld and N. Killoran. Quantum machine learning in feature hilbert spaces. *Physical review letters*, 122(4):040504, 2019.

M. Schuld and F. Petruccione. *Machine Learning with Quantum Computers*. Springer, 2021.

M. Schuld, I. Sinayskiy, and F. Petruccione. An introduction to quantum machine learning. *Contemporary Physics*, 56(2):172–185, 2015.

A. Shaban, M. Farajtabar, B. Xie, L. Song, and B. Boots. Learning latent variable models by improving spectral solutions with exterior point method. In *UAI*, pages 792–801, 2015.

L. A. Shepp and Y. Vardi. Maximum Likelihood Reconstruction for Emission Tomography. *IEEE Transactions on Medical Imaging*, 1(2):113–122, Oct. 1982. ISSN 1558-254X. doi: 10.1109/TMI.1982.4307558.

S. Siddiqi, B. Boots, and G. Gordon. Reduced-rank hidden markov models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 741–748. JMLR Workshop and Conference Proceedings, 2010.

S. Singh, M. R. James, and M. R. Rudary. Predictive state representations: A new theory for modeling dynamical systems. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, UAI '04, pages 512–519, Arlington, Virginia, United States, 2004. AUAI Press. ISBN 0-9749039-0-6. URL `http://dl.acm.org/citation.cfm?id=1036843.1036905`.

A. Smola, A. Gretton, L. Song, and B. Schölkopf. A hilbert space embedding for distributions. In *International Conference on Algorithmic Learning Theory*, pages 13–31. Springer, 2007.

J. A. Smolin, J. M. Gambetta, and G. Smith. Efficient Method for Computing the Maximum-Likelihood Quantum State from Measurements with Additive Gaussian Noise. *Physical Review Letters*, 108(7):070502, Feb. 2012. doi: 10.1103/PhysRevLett.108.070502.

L. Song, J. Huang, A. Smola, and K. Fukumizu. Hilbert space embeddings of conditional distributions with applications to dynamical systems. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 961–968, 2009.

L. Song, B. Boots, S. Siddiqi, G. J. Gordon, and A. Smola. Hilbert space embeddings of hidden markov models. 2010.

L. Song, K. Fukumizu, and A. Gretton. Kernel embeddings of conditional distributions: A unified kernel framework for nonparametric inference in graphical models. *IEEE Signal Processing Magazine*, 30(4):98–111, 2013.

S. Srinivasan and S. Adhikary. Learning physically valid quantum tensor networks, 2021.

S. Srinivasan, C. Downey, and B. Boots. Learning and Inference in Hilbert space with Quantum Graphical Models. In *Advances in Neural Information Processing Systems 31*, 2018a.

S. Srinivasan, G. Gordon, and B. Boots. Learning hidden quantum markov models. In *International Conference on Artificial Intelligence and Statistics*, pages 1979–1987, 2018b.

S. Srinivasan, S. Adhikary, J. Miller, G. Rabusseau, and B. Boots. Quantum tensor networks, stochastic processes, and weighted automata. *arXiv preprint arXiv:2010.10653*, 2020.

S. Srinivasan, S. Adhikary, J. Miller, B. Pokharel, G. Rabusseau, and B. Boots. Towards a trace-preserving tensor network representation of quantum channels. *Second Workshop on Quantum Tensor Networks in Machine Learning (NeurIPS)*, 2021.

J. Stokes and J. Terilla. Probabilistic modeling with matrix product states. *Entropy*, 21 (12):1236, 2019.

E. Stoudenmire and D. J. Schwab. Supervised learning with tensor networks. *Advances in Neural Information Processing Systems*, 29, 2016.

T. Surawy-Stepney, J. Kahn, R. Kueng, and M. Guta. Projected least-squares quantum process tomography. *arXiv preprint arXiv:2107.01060*, 2021.

M. Thon and H. Jaeger. Links between multiplicity automata, observable operator models and predictive state representations — a unified learning framework. *Journal of Machine Learning Research*, 16:103–147, 2015. ISSN 15337928.

G. Torlai, C. J. Wood, A. Acharya, G. Carleo, J. Carrasquilla, and L. Aolita. Quantum process tomography with unsupervised learning and tensor networks. *arXiv preprint arXiv:2006.02424*, 2020.

G. G. Towell, M. O. Noordewier, and J. W. Shavlik. Molecular biology (splice-junction gene sequences) data set, 1991. "Available at `https://archive.ics.uci.edu/ml/datasets`".

L. Vanderstraeten, J. Haegeman, and F. Verstraete. Tangent-space methods for uniform matrix product states. *SciPost Physics Lecture Notes*, Jan 2019. ISSN 2590-1990. doi: 10.21468/scipostphyslectnotes.7. URL `http://dx.doi.org/10.21468/SciPostPhysLectNotes.7`.

M. Vidyasagar. The complete realization problem for hidden markov models: a survey and some new results. *Mathematics of Control, Signals, and Systems*, 23(1-3):1–65, 2011.

I. Volobouev. On the Expectation-Maximization Unfolding with Smoothing, Jan. 2015. URL `http://arxiv.org/abs/1408.6500`.

Z. Wen and W. Yin. A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, 142(1-2):397–434, 2013a. ISSN 00255610. doi: 10.1007/s10107-012-0584-1.

Z. Wen and W. Yin. A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, 142(1-2):397–434, 2013b.

A. H. Werner, D. Jaschke, P. Silvi, M. Kliesch, T. Calarco, J. Eisert, and S. Montangero. Positive tensor network approach for simulating open quantum many-body systems. *Physical review letters*, 116(23):237201, 2016.

S. R. White. Density matrix formulation for quantum renormalization groups. *Phys. Rev. Lett.*, 69:2863–2866, Nov 1992. doi: 10.1103/PhysRevLett.69.2863. URL `https://link.aps.org/doi/10.1103/PhysRevLett.69.2863`.

E. W. Wiewiora. *Modeling probability distributions with predictive state representations*. PhD thesis, UC San Diego, 2008.

C. J. Wood, J. D. Biamonte, and D. G. Cory. Tensor networks and graphical calculus for open quantum systems. *Quantum Info. Comput.*, 15(9-10):759–811, July 2015. ISSN 1533-7146. URL `http://dl.acm.org/citation.cfm?id=2871422.2871425`.

K. Wright, K. M. Beck, S. Debnath, J. M. Amini, Y. Nam, N. Grzesiak, J.-S. Chen, N. C. Pisenti, M. Chmielewski, C. Collins, K. M. Hudek, J. Mizrahi, J. D. Wong-Campos, S. Allen, J. Apisdorf, P. Solomon, M. Williams, A. M. Ducore, A. Blinov, S. M. Kreikemeier, V. Chaplin, M. Keesan, C. Monroe, and J. Kim. Benchmarking an 11-qubit quantum computer. *Nature Communications*, 10(1):5464, Nov. 2019. ISSN 2041-1723. doi: 10.1038/s41467-019-13534-2.

B. Yang, R. Raymond, and S. Uno. An Efficient Quantum Readout Error Mitigation for Sparse Measurement Outcomes of Near-term Quantum Devices, Feb. 2022. URL `http://arxiv.org/abs/2201.11046`.

R. Yu, S. Zheng, A. Anandkumar, and Y. Yue. Long-term forecasting using higher order tensor rnns. *arXiv: Learning*, 2017.

M. Zhao and H. Jaeger. Norm observable operator models. Technical report, Jacobs University Bremen, 2007.

M.-J. Zhao and H. Jaeger. Norm Observable Operator Models. *Neural Computation*, 22 (7):1927–1959, 2010.

K. Życzkowski and I. Bengtsson. On Duality between Quantum Maps and Quantum States. *Open Systems & Information Dynamics*, 11(1):3–42, 2004. ISSN 1230-1612, 1573-1324. doi: 10.1023/B:OPSY.0000024753.05661.c2.

Appendix A

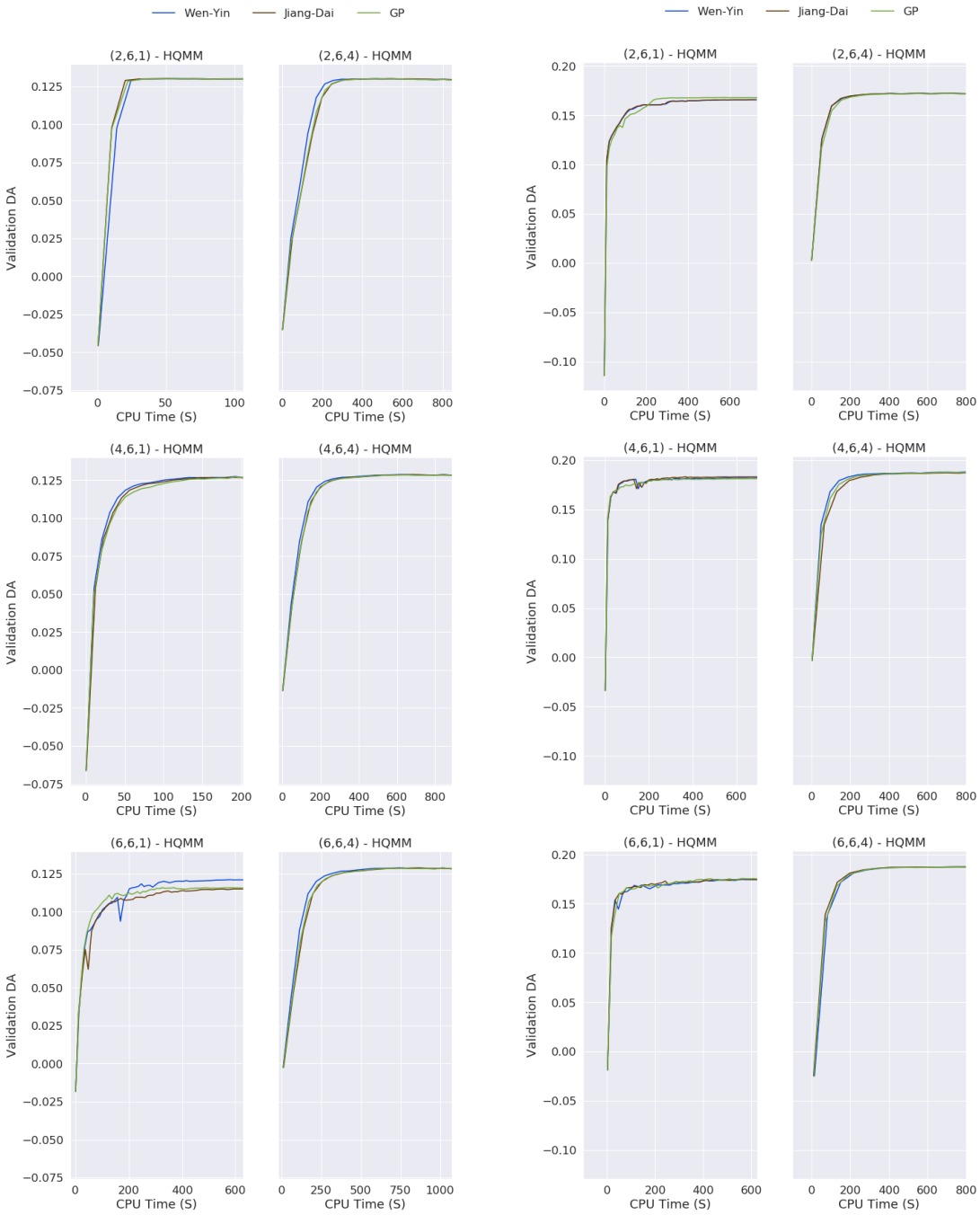
Appendix

A.1 Additional Details on HQMM Experiments

A.1.1 Update Schemes on the Stiefel Manifold

Algorithms that constrain parameters on the Stiefel manifold generally are either projection-like (which re-orthogonalize the naive gradient descent updates) or geodesic-like (which directly generate updates on the manifold itself). Among geodesic-like algorithms, those proposed by Wen and Yin [2013a] and Jiang and Dai [2013] are the current state-of-the-art approaches. In the regime of tall-and-skinny matrices in our problem, these two are theoretically equivalent and have the same computational complexity $O(7Nn^2)$. By comparison, the canonical gradient projection algorithm has a slightly lower computational complexity of $O(3Nn^2)$ [Jiang and Dai, 2013]. We compared these three update schemes to project or retract gradients onto the Stiefel manifold. The exact update schemes for all three methods can be found in Jiang and Dai [2013].

We trained 9 HQMM models for both the synthetic HQMM and synthetic HMM datasets using these 3 update schemes. As shown in the results in Figures A.1a and A.1b, the three methods are very similar both in terms of speed and the final solution quality for our benchmark datasets. Since the Wen-Yin update was slightly faster, especially for larger models on the synthetic HQMM data, we used it over the alternatives.



(a) Results for the Synthetic HQMM Data

(b) Results for the Synthetic HMM Data

Figure A.1: Alternative Schemes to Constrain Updates on the Stiefel Manifold Validation set accuracies obtained for HQMMs trained using different update schemes. All schemes provide similar speed and model qualities, but the Wen-Yin update outperforms the others by a small margin.

A.1.2 Sensitivity to Initialization

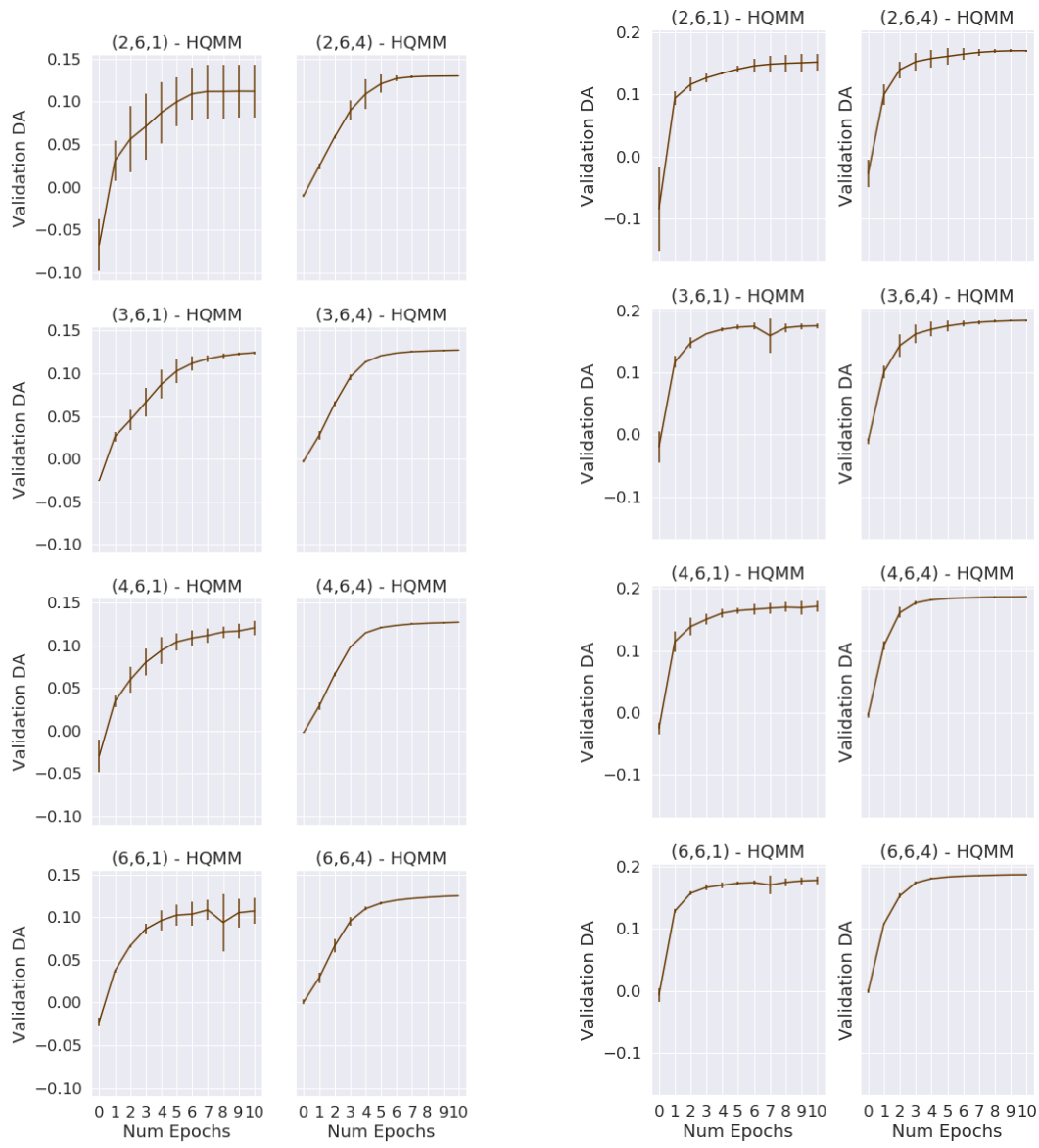
The COSM algorithm begins with an initial guess of the optimal parameters κ and a random intial density matrix $\hat{\rho}$. By 'burning-in' a reasonable number of initial entries in sequences, we minimize the effect of randomly initializing $\hat{\rho}$. To investigate the sensitivity of COSM to initializations of κ, we trained models on the synthetic HQMM and HMM datasets over 3 random seeds. As shown in the results in Figure A.2a and A.2b, COSM is sensitive to random initializations for the smallest $(2, 6, 1)$ model, but the variance in DA scores quickly decrease with an increase in model size, both as a function of n and w. We observe even lower variance across different initializations for the synthetic HMM data in Figure A.2b.

A.1.3 Hyperparameter Selection

To facilitate a clear comparison with the GS approach to learning HQMMs [Srinivasan et al., 2018b], we used the same batch size as in Srinivasan et al. [2018b], and tuned the step-size τ and decay rate α for all HQMM models. We started by manually tuning models, and identified that all models tended to converge to good solutions with the following hyperparameters: $\tau = 0.75$ and $\alpha = 0.92$ for the synthetic datasets, and $\tau = 0.8$ and $\alpha = 0.9$ for the splice dataset. We trained baseline models using these parameters, and then randomly searched for better configurations around these values.

For the synthetic datasets, we fixed the batch size at 20 and randomly sampled τ between 0.55 and 0.95, and α between 0.9 and 0.99. As we wanted to explore many hyperparameter settings, we only trained on 3 random batches in every epoch. For the splice dataset, we fixed the batch size at 200 and randomly sampled τ between 0.7 and 0.9 and α between 0.88 and 0.92. Since each splice model required learning three separate HQMMs across multiple folds, we tested fewer hyperparameter settings across a smaller search space. We also trained on a single random batch every epoch across 2 folds.

Given the large number of models that we needed to evaluate, we used the Hyperband scheduling technique [Li et al., 2017] to quickly sample through many hyperparameter



(a) Synthetic HQMM Data

(b) Synthetic HMM Data

Figure A.2: COSM's Sensitivity to Random Initializations of κ Validation set accuracies obtained across 10 epochs for HQMMs trained on 3 different random initializations. COSM is sensitive to κ initialization for the smallest models, but is fairly robust for larger models.

configurations. For each model, we began by running 3 epochs for each of the k randomly selected configurations, and removed $k/3$ of them with the lowest validation DA scores. In the next round, we ran the remaining configurations for a larger number of iterations, and again removed the bottom third of the configurations with the lowest scores. We repeated

Table A.1: Hyperparameter Selection The best performing step sizes (τ) and decay rates (α) for various COSM models. For models not listed here, the default hyperparameters ($\tau = 0.75, \alpha = 0.92$) and ($\tau = 0.8, \alpha = 0.9$) yielded the best results for the synthetic datasets and the splice dataset respectively.

Dataset	n	s	w	τ	α
Synthetic HQMM	2	6	1	0.75	0.92
5*Synthetic HMM	2	6	1	0.95	0.99
	4	6	6	0.95	0.96
	5	6	1	0.55	0.96
	5	6	2	0.95	0.98
	5	6	6	0.95	0.99
7*Splice	2	4	1	0.70	0.90
	2	4	2	0.85	0.92
	2	4	6	0.85	0.92
	4	4	1	0.90	0.92
	4	4	4	0.90	0.90
	6	4	4	0.70	0.90
	8	4	1	0.90	0.90

this strategy until only one configuration remained, and saved the one with the highest validation DA throughout the tuning protocol. We searched across 27 and 9 random configurations for the synthetic and the splice datasets respectively. As an example, for the synthetic datasets we would train 27 models for 3 epochs, followed by the 9 best models for 9 epochs, followed by the 3 best models for 9 epochs, and the final best model for 27 epochs. In Table A.1, we report the hyperparameters obtained through Hyperband that outperformed the default configuration. For models not listed in the table, the default configuration resulted in the best performance.

All our experiments were performed on a desktop with 8 Intel Core i7-7700K 4.20 GHz CPUs, and 31.3 GB RAM. All models are trained in MATLAB, but the gradient computation happens in Python.

A.1.4 COSM's Speedup over GS

Since the GS method can take days to converge to the final solution for larger models such as $(6, 6, 6)$-HQMMs, it was not feasible to compute a direct speed up comparing its convergence time to COSM across most models. Thus, we estimate the speed-up offered

by COSM by fitting a linear model to the DA trajectory of models learned by the GS method. Specifically, for a given HQMM model, we train both COSM and GS on the synthetic HMM data until one of them converges within a tolerance of 10^{-5} in DA scores. Since COSM always converges first, we take the DA scores achieved by GS in its last 10 steps and fit a linear model to it. We then extrapolate this linear model to estimate the time it would take for GS to reach some fraction of the solution DA reached by COSM. Note that a linear fit is an optimistic assumption of GS convergence time, meaning we are going to *understate* how much faster COSM is compared to GS. Finally, we estimate the speed up offered by COSM as the ratio of the (estimated) convergence time for GS and the actual convergence time for COSM. In Figure A.3, we plot this estimated speed up with varying number of parameters (both as functions of n and w) for different solution fractions. For a solution fraction of 1, we record speedups greater than $150\times$ for the largest HQMMs trained. Furthermore, COSM offers comparable increase in speed up as parameters grow either by virtue of increasing the number of latent states n or the auxiliary dimension w.

A.2 IBM Device Specifications in MEM Experiments

We give device specifications in Table A.2. We show the device architectures in Figures A.4 and A.5.

	Montreal			**Washington**		
	Min	Mean	Max	Min	Mean	Max
T_1 (μs)	57.57	113.2	187.1	40.43	101.98	247.7
T_2 (μs)	22.58	99.72	198.71	2.61	94.29	259.39
1QG Error (%)	0.02	0.04	0.15	0.02	0.08	1.09
2QG Error (%)	0.57	1.35	6.09	0.53	4.79	100.
1QG Duration (μs)	0.04	0.04	0.04	0.04	0.04	0.04
2QG Duration (μs)	0.27	0.43	0.63	0.27	0.55	1.34
RO Error (%)	0.79	2.59	10.66	0.23	3.51	38.49
RO Duration (μs)	5.2	5.2	5.2	0.86	0.86	0.86

Table A.2: Device specifications for Montreal and Washington. 1QG and 2QG denote 1-qubit gate and 2-qubit gate, respectively. RO denotes readout.

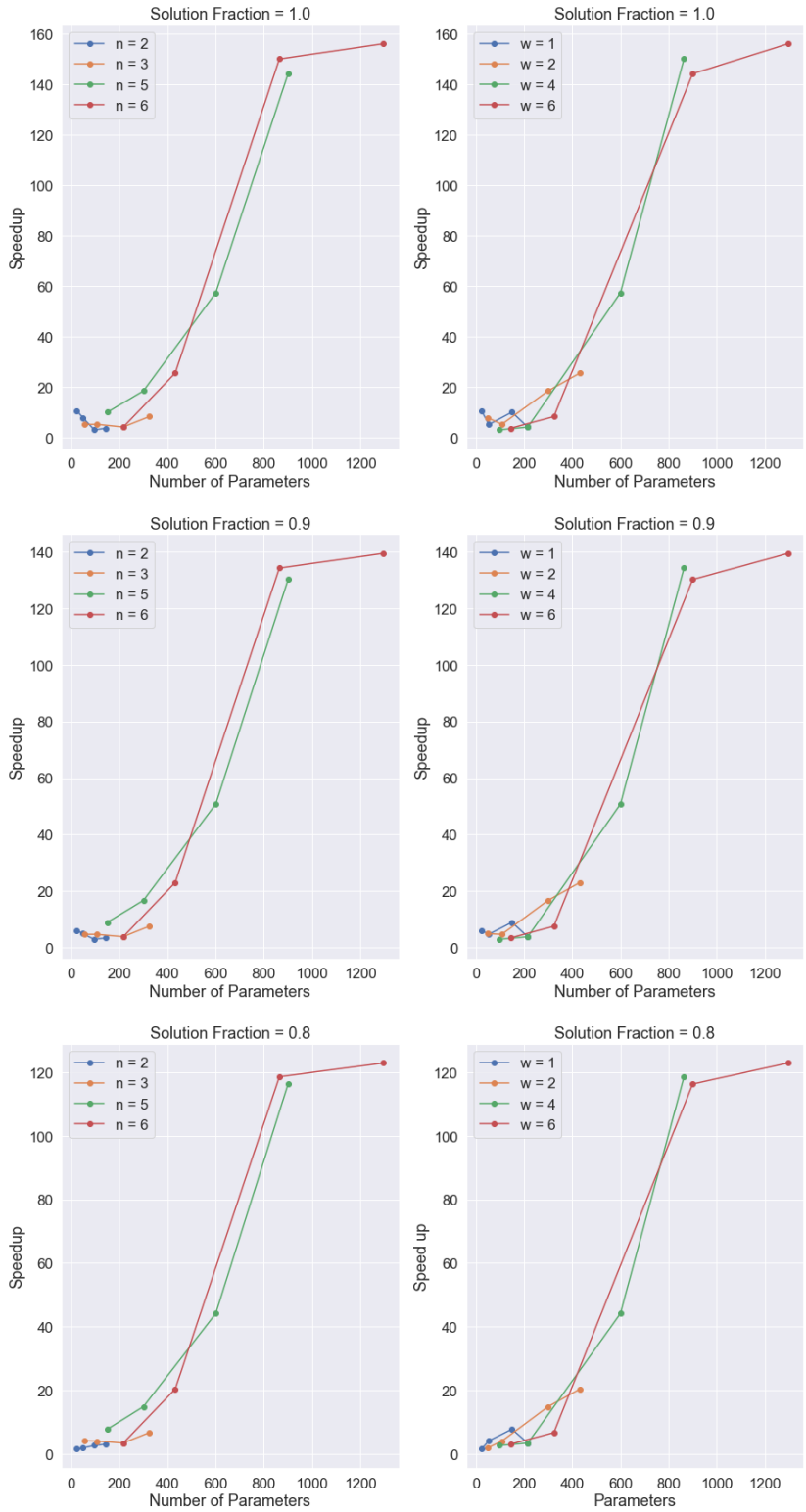


Figure A.3: Estimated Speedup of COSM over GS: Estimated speedups of COSM over GS for various solution fractions. As seen in the plots for solution fraction of 1, GS can take more than 150 times the convergence time for COSM to reach the latter's final solution quality.

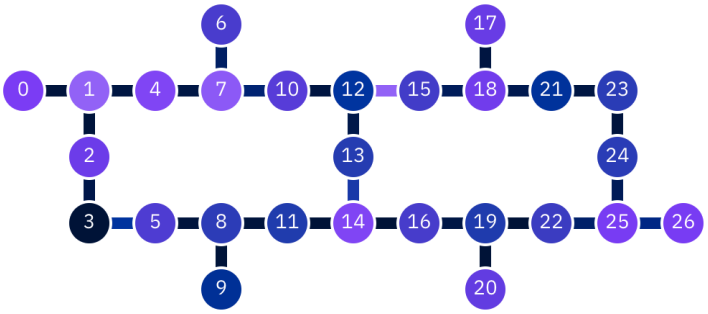


Figure A.4: Schematic for the device connectivity for IBMQ Montreal. The entire device was used to implement 26-qubit Bernstein-Vazirani algorithm.

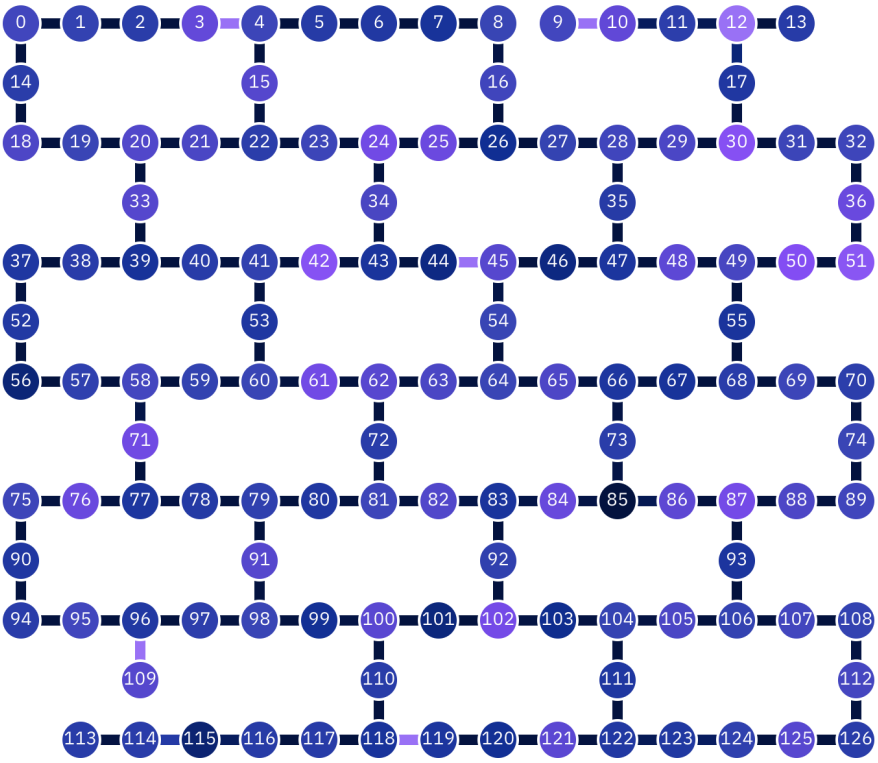


Figure A.5: Schematic for the device connectivity for IBMQ Washington. We constructed GHZ(n) states for $n = 2$ to $n = 127$. However, the 'correct' bitstrings (all 0s and all 1s) were no longer the modal measured bitstrings beyond $n = 81$ qubits, and the normalized ℓ_1 score was 0 (see Figure 4.6).