Mind the Metric: Methods in Metric-Informed Machine Learning

Sandesh Adhikary

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

Doctor of Philosophy

University of Washington

2025

Reading Committee:

Byron Emereth Boots, Chair

Abhishekh Gupta

Sewoong Oh

Program Authorized to Offer Degree:
Computer Science and Engineering

University of Washington

Abstract

Mind the Metric: Methods in Metric-Informed Machine Learning

Sandesh Adhikary

Chair of the Supervisory Committee:
Byron Emereth Boots
Computer Science and Engineering

Various machine learning algorithms can benefit from exploiting the metric structure inherent in data and model parameters. Such geometric structure can be a useful prior to reduce sample complexity, improve generalization, and ensure feasibility of learned models. We present a range of metric-informed machine learning methods that span probabilistic graphical models, reinforcement learning, and sampling from probability distributions.

Within the domain of **probabilistic graphical models**, we use metric structure to learn feasible quantum-graphical models using geometry-aware optimization. Specifically, we present an approach to learn hidden quantum Markov models (HQMMs) through Riemannian gradient descent on the Stiefel manifold. Moreover, we establish a hierarchy of expressiveness relationships between HQMMs and linear sequential systems from various fields including machine learning, formal language, and quantum information.

Within the domain of **sampling based probabilistic inference**, we use metric structure to define geometry-aware similarity measures for sampling in structured data spaces. Specifically, we adapt the kernel herding algorithm to non-Euclidean spaces using geometry-aware kernels and Riemannian optimization. We demonstrate through empirical evaluations that Riemannian kernel herding outperforms various common heuristics and other geometry-informed approaches.

Within the domain of **reinforcement learning** (RL), we use metric structure to define geometry-aware representations of states and policies. Specifically, we introduce Behavioral

Eigenmaps (BeigeMaps) – state representations that preserve the local metric structure induced by bisimulation metrics through neural eigendecompositions of similarity kernels. When added as a drop-in modification, BeigeMaps improve the policy performance of prior behavioral distance-based RL algorithms by highlighting value-based clusters in state spaces. Moreover, we also introduce a framework for policy composition to aggregate policies learned in multi-objective RL as policy-centroids in distance spaces where policies are embedded.

ACKNOWLEDGMENTS

This dissertation would not have been possible without the academic and emotional support from family, friends, and colleagues. While any errors in this thesis are entirely my own, any positive progress made through this dissertation has been a collective effort of all those who helped me through the process.

Firstly, I would like to thank my advisor Byron Boots, and my committee members including Sewoong Oh, Abhishek Gupta, and Yen-Chi Chen. I am grateful for the support they provided to shape this thesis into its ultimate form. I am particularly indebted to Byron for his guidance throughout my PhD. I first came across Byron and his work early in my PhD when I felt lost as a researcher. His encouragement and openness to exploring new ideas outside his comfort zone rejuvenated my interest in research. I deeply appreciate the confidence Byron placed in me to let me freely explore ideas from quantum information to robotics, with many enjoyable stops in between.

I would also like to thank the great research collaborators I have had the pleasure of working with throughout my PhD including Siddarth Srinivasan, Bibek Pokharel, Jacob Miller, Anqi Li, Jacob Sacks, Josie Thompson, Tyler Westenbroek, Chandra Bhagavatula, Geoff Gordon and Guillaume Rabusseau. I am particularly grateful to Siddarth for collaborating with me on much of the quantum-inspired machine learning research presented in this thesis. I have learned a lot about research, writing, and productive collaboration through our work together. It has taken a village to produce the work in this dissertation, and I am grateful to have been in such great company throughout the process.

Finally, I would like to thank my family and friends who have been a constant source of moral and emotional support. Thank you to everyone at the Robot Learning Lab at UW and Georgia Tech for being such great friends and collaborators – including Jake, Sid, Anqi, Mohak, Rosario, Rohan, Mateo, Carolina, Tyler, Kevin, Sanghun, Sasha, Nolan, and

Adam. Thank you to Ashwin and Nikesh for always being there through thick and thin. I am immensely grateful to my parents and my sister Pragya for their constant support and encouragement. Finally, thank you to my wife Nisma Elias, who has been my rock throughout this PhD and beyond. I feel so lucky to have you as a companion in this journey – having you by my side has made difficulties more endurable and successes all the more joyous. Thank you!

Sandesh Adhikary

Mind the Metric: Methods in Metric-Informed Machine Learning

PHD. DISSERTATION

Dept. of Computer Science & Engineering
University of Washington,
Seattle, WA

Contents

List of Figures

List of Tables

Part I

Introduction

1
Methods in Metric Informed Machine Learning

1.1 Introduction

Incorporating structural priors into machine learning algorithms is a powerful way to improve their generalization and overall performance. The central hypothesis of such an approach is that instead of relying on the algorithm to infer structural constraints from data, it may be more efficient to inject the structure into the algorithm itself. Exploiting the structure of the problem can accelerate learning by inducing useful inductive biases and regularization, especially in problems with limited or noisy data. This structure can take a variety of forms including the physics dictating a system (Raissi et al., 2019; Wu et al., 2017; Karniadakis et al., 2021), the form of underlying dynamical systems (Moerland et al., 2023; Rawlings, 2000; Greydanus et al., 2019), causal relationships between variables (Schölkopf, 2022), organization of semantic relationships in hierarchies or groups (Scarselli et al., 2008; Hutsebaut-Buysse et al., 2022; Mikolov et al., 2013), and so on. We focus on a particular form of structure, namely metric structure.

A metric, in essence, is simply a way to quantify similarity – or more precisely dissimilarity – between entities, which in machine learning generally correspond to data or model parameters[1]. The choice of the metric induces a metric structure that introduces structural constraints such as curvature, symmetry, invariance, and feasibility restrictions for optimization. Thus, choosing the right metric for a problem and adapting an algorithm to respect this choice provides a way to inject structural priors. Recently, the field of geometric learning, and especially geometric deep learning, has not only produced various approaches to align models with metric constraints of input spaces, but also provided insight on how many successful machine learning algorithms can be viewed as instantiations of geometric learning (Bronstein et al., 2021). For instance, the remarkably powerful class of convolutional neural networks can be understood

[1] We provide a formal introduction to metrics in Chapter 2

as a specific architecture that ensures the geometric notion of translational invariance (Bronstein et al., 2021). Insights such as this have led to the development of many alternatives utilizing a wider range of invariance and equivariance properties (Cohen and Welling, 2016; Finzi et al., 2020, 2021; Cohen et al., 2019). Moreover, metric-informed methods have been foundational in various classical methods in unsupervised learning including clustering (Von Luxburg, 2007), dimensionality reduction (Tenenbaum et al., 2000; Belkin and Niyogi, 2001; Roweis and Saul, 2000), manifold learning (Meilă and Zhang, 2024), and so on. Overall, exploiting metric structure has been a powerful and successful approach for integrating structure in machine learning.

In this dissertation, we essentially tackle the following question: *How can we utilize metric structure in machine learning algorithms to improve their performance*? However, this is admittedly too broad a question within the scope of the work presented in this dissertation. Indeed, the entire literature on metric-informed machine learning is essentially an attempt to answer this question. Limiting the scope of our objective, we focus on selected problems in machine learning by making specific choices for the three main components of the question: the metric structure, the target algorithm, and the type of improvement expected by incorporating the metric structure. Our work thus builds on the growing body of literature in metric-informed machine learning, specifically demonstrating its applicability in specific problems in three domains: probabilistic graphical modeling, sampling from probability distributions, and reinforcement learning. We now discuss the role of metrics in these domains, and define the specific questions we aim to address.

Metric-Informed Probabilistic Modeling Metric structure plays a crucial role in probabilistic modeling, as the space of probability distributions itself is endowed with rich geometric properties. This space, often referred to as the statistical manifold (Amari, 1987), is formalized in the field of information geometry using tools from differential geometry (Amari, 2016; Nielsen, 2020). Here, the Fisher information matrix serves as a canonical Riemannian metric that defines the properties of the statistical manifold. Beyond intrinsic metrics, several extrinsic distances between probability distributions are also fundamental tools in machine learning, offering various ways to quantify dissimilarity between distributions. Common examples include the Kullback-Leibler (KL) divergence, total variation distance, Rényi divergence, Hellinger distance, and chi-squared divergence. Each of these measures has unique properties that make it suitable for different tasks, depending on the structure of the data and spe-

cific needs of the application.

Adding to the toolkit of probability metrics is another rich class of metrics and distances that leverage the structure of the underlying data spaces. For instance, the maximum mean discrepancy (MMD) (Gretton et al., 2012) defines distances between distributions embedded in reproducing kernel Hilbert spaces and has been widely applied in two-sample testing, generative modeling (Li et al., 2017a), domain adaptation (Quiñonero-Candela et al., 2008), and more. Similarly, the Wasserstein distance, which forms the foundation of optimal transport theory (Monge, 1781; Peyré et al., 2019; Kantorovich, 2006), captures geometric discrepancies between distributions and is extensively used in shape analysis, generative modeling, and structured prediction. Sinkhorn distances provide computationally efficient approximations to Wasserstein distances, making them practical for large-scale optimization problems (Cover, 1999; Cuturi and Doucet, 2014).

In addition to utilizing the geometry of the statistical manifold, defining probability distributions over structured geometric spaces, such as Riemannian manifolds, is also crucial for accurately modeling data that inherently resides on curved or constrained domains. Classical Euclidean assumptions can be inadequate for data on non-Euclidean spaces like spheres (e.g., directions and orientations), rotation groups (e.g., rigid body transformations), or the space of symmetric positive definite (SPD) matrices (e.g., covariance matrices in brain imaging or finance). Statistics on manifolds provides tools for extending notions like means, variances, and distributions to these spaces, enabling principled learning and inference (Pennec, 2006). The von Mises-Fisher (Fisher, 1953) and Bingham distributions (Bingham, 1974) are foundational in directional statistics (Mardia and Jupp, 2009). Riemannian normal distributions on SPD manifolds support tasks in robotics (Yilmaz and Shah, 2008; Barfoot, 2024), finance (Ledoit and Wolf, 2003), diffusion tensor imaging (Pennec et al., 2006), and so on. In robotics, distributions over the special Euclidean group $SE(n)$ and the Special orthongonal group $SO(n)$ are used for probabilistic modeling and inference on the space of rigid-body transformations, rotations, and orientations (Barfoot, 2024). By accounting for the geometry of the underlying space, these approaches enable accurate and geometrically consistent models than their Euclidean counterparts.

Within the domain of probabilistic graphical models, we utilize metrics to perform geometry-aware optimization that ensures the feasibility of a specific class of probabilistic graphical models learned from data. Specifically, we focus on quantum-inspired generalizations of hidden Markov models (HMMs) known as hidden quantum

Markov models (HQMMs). As we show in subsequent chapters, HQMMs form a broadly expressive model class for linear stochastic processes, and the constraints placed on its model parameters ensure feasible probabilistic modeling. This is in contrast to some other classical generalizations of HMMs that suffer from the well-known negative-probability problem (Wiewiora, 2007), and consequently require heuristic projections or approximations in learning algorithms. By characterizing the metric structure of HQMMs as the Stiefel manifold, we present a learning algorithm that exploits this structure via Riemannian gradient descent. Furthermore, we also develop a hierarchy of expressiveness comparing how these constraints differentiate these models from numerous other linear sequential systems from the machine learning community (i.e. observable operator models (Jaeger, 2000), predictive state representations (Singh et al., 2004), tensor trains (Oseledets, 2011)), formal language community (i.e., stochastic weighted automata (Balle et al., 2014a)), and quantum information (i.e., quantum tensor networks (Biamonte and Bergholm, 2017)). In summary, we tackle the following research question in Part 2 of this dissertation:

Part 2: Metric Informed Quantum Graphical Models:
How can we utilize *the Stiefel manifold* structure of model parameters in *hidden Quantum Markov models* to ensure *feasibility and improve accuracy*.

Metric-Informed Sampling Sampling from probability distributions is a central problem in Bayesian inference, and the information geometry of the space of probability distributions has been crucial in analyzing and improving existing sampling algorithms. Viewing the parameter space of statistical models as a Riemannian manifold, Girolami and Calderhead (2011) have proposed Riemannian generalizations of Markov chain Monte Carlo (MCMC) algorithms including the Metropolis adjusted Langevin algorithm and Hamiltonian Monte Carlo (Girolami and Calderhead, 2011), which are particularly useful when target distributions exhibit strong correlations or are defined over high-dimensional data spaces. Another line of work in particle-based variational inference treats the sampling problem as transporting a set of samples towards target distributions. The Stein variational gradient descent (SVGD) algorithm (Liu and Wang, 2016) is a prototypical example of such an approach, which simulates the gradient flow of the KL-divergence with respect to a reproducing kernel Hilbert space (Duncan et al., 2023; Liu, 2017). Underlying this method is the kernelized Stein discrepancy (KSD) (Chwialkowski et al., 2016; Gorham and Mackey, 2015), a kernel

based distance measure between distributions. Alternative methods utilizing the same geometry and KSD have also been proposed, such as the Stein points algorithm (Chen et al., 2018, 2019). Both SVGD and Stein points allow sampling from distributions without requiring computationally intractable normalizations, and also prevent positive autocorrelations often observed in MCMC methods. Moreover, other probability metrics have also been used to design alternative sampling approaches such as the kernel herding algorithm (Chen et al., 2010) that minimizes the maximum mean discrepancy between kernel mean embeddings of distributions (Muandet et al., 2017).

Besides the information geometry of the space of probability distributions, the geometry of the sample space itself is also of interest in sampling. When the data space is required to satisfy geometric constraints, we require generated samples to be feasible with respect to these constraints. There has also been literature dedicated to extending Markov Chain Monte Carlo (MCMC) methods to Riemannian manifolds (Brubaker et al., 2012; Byrne and Girolami, 2013; Diaconis et al., 2013). De Bortoli et al. (2022) introduce Riemannian score-based generative models that expand the original generative algorithm to Riemannian manifolds. The Riemannian extension of Stein variational gradient descent (Liu and Zhu, 2018) provides a particle-based approach to sampling from unnormalized distributions defiend over Riemannian manifolds.

Within the domain of sampling, we utilize metrics to define geometry-aware similarities to draw samples on structured data spaces. Specifically, we present an approach to adapting the kernel-herding algorithm to Riemannian manifolds. Known for the fast convergence rate of the 'super samples' it generates, kernel herding is an algorithm to draw samples from probability distributions via their kernel-mean embeddings in a reproducing kernel Hilbert space (Chen et al., 2010). However, standard applications of the algorithm do not immediately translate when samples must lie on Riemannian manifolds. We propose to adapt kernel herding to Riemannian manifolds by using geometry-aware kernels that incorporate the appropriate distance metric for the manifold, and Riemannian optimization to constrain herded samples to lie on the manifold. In summary, we tackle the following research question in Part 3 of this dissertation:

Part 3: Metric Informed Sampling
How can we utilize *the Riemmanian manifold* structure of data spaces in *kernel herding* to ensure *feasibility of samples and improve convergence to target distributions*.

Metric-Informed Reinforcement Learning and Control A central problem in reinforcement learning is to reduce the sample complexity of learning optimal policies (Kakade, 2003; Lattimore et al., 2013; Zhang et al., 2024). Unlike most supervised and unsupervised learning algorithms, reinforcement learning algorithms need to incrementally construct datasets through repeated and potentially time-consuming interactions with an unknown environment. Therefore, making efficient use of data is particularly important – especially in environments with high-dimensional observations such as images. Given the complexity of learning optimal behaviors in changing environments in general, data efficiency is similarly important for control algorithms beyond reinforcement learning as well. Consequently, there is a rich literature of approaches to improve the efficiency of RL and control algorithms by exploiting the geometry inherent in various components of the underlying problem, including in state spaces, transition dynamics, policy spaces, and the space of value functions.

When the state space of a control problem exhibits explicit geometric structure, it is clear that the induced geometric constraints must be satisfied to ensure feasibility. For instance, adapting probabilistic models to the Riemannian geometry of the state-space can allow design of principled control algorithms (Calinon, 2020; Saveriano et al., 2023; Zhu, 2014), as well as geometry-aware Bayesian inference and optimization (Jaquier et al., 2020, 2022). But even when the state space is ostensibly unstructured, it is still possible to construct useful state representations that capture the geometry of state space dynamics. In reinforcement learning, a prototypical example is that of proto value-functions (PVFs) – state representations corresponding to Fourier eigenfunctions of the Laplace-Beltrami diffusion operator on the state space manifold (Mahadevan, 2005). For value functions corresponding to arbitrary rewards, PVFs provide a set of basis functions that are geodesically smooth with respect to the manifold induced by state-space dynamics. This idea has been extended in various directions, including Krylov subspace methods (Petrik, 2007; Ghosh and Bellemare, 2020), singular value decomposition techniques (Duan et al., 2019), spectral graph drawing (Wu et al., 2018; Wang et al., 2021), and action-aware representations (Ren et al., 2022), among others (Ghosh and Bellemare, 2020). Notably, PVFs also align with the scaled eigenvectors of successor representations (Dayan, 1993; Stachenfeld et al., 2014). The EigenOptions framework (Machado et al., 2017a,b) leverages this relationship for both option discovery and representation learning.

Besides exploiting the structure of the state space, various works also utilize the inherent geometry of the space of policies. The central insight behind the popular natural policy gradient algorithm

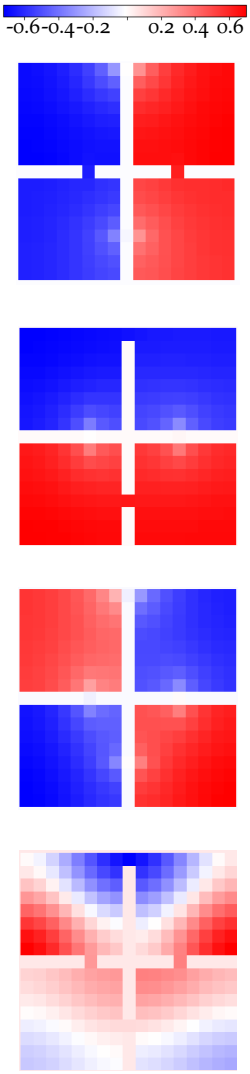


Figure 1.1: **Proto Value Functions on a Discrete 4-room Environment** The first four proto-value functions (PVFs) in a simple discrete 4-room environment, where each room is connected to its neighbor by a narrow corridor. The PVFs provide a basis for the smooth flow of value in the environment for arbitrary reward functions, with the first three encoding flows from left to right, bottom to top, and diagonal respectively, while accounting for bottlenecks from corridors. Higher order PVFs encode less smooth flows.

(Kakade, 2001) is to define a metric based on the underlying structure of the policy space – the Fisher metric, which is invariant on the space of the parameters of probability distributions. The Stein variational policy gradient algorithm (Liu et al., 2017) utilizes Stein geometry via SVGD to optimize particles to match policy distributions. In robotics, Riemannian motion policies allow decomposing a complex robot into multiple constituent controllers which are composed by translating between their individual Riemannian metrics (Cheng et al., 2021; Li et al., 2021). This framework has also been expanded further into geometric fabrics utilizing the broader class of Finsler geometry (Van Wyk et al., 2022). Various policy-composition strategies in multi-objective reinforcement learning can also be viewed as averages over distances spaces induced by policies (Adhikary and Boots, 2022).

Rich metric structure also exists within the space of value functions in reinforcement learning problems. Bellemare et al. (2019) utilize the geometric properties of the space of value functions to formally establish the usefulness of value functions as auxiliary tasks. Furthermore, the bisimulation (pseudo) metric defines a notion of behavioral similarity between states that capture their similarities based on current reward, as well as future transitions under optimal or arbitrary policies (Ferns et al., 2011; Castro, 2020). These metrics can be learned from data in conjunction with policies to induce state representations that reveal natural clusters with respect to the value function. In high dimensional state spaces such as images, this can provide a useful prior to make policies robust to irrelevant distractors and more sample efficient (Zhang et al., 2020; Chen and Pan, 2022; Adhikary et al., 2024).

Within the domain of reinforcement learning, we utilize metrics to define geometry-aware representations of states and policies. Firstly, we devise an algorithm to encode bisimulation metrics into state space representations that allow long-range metric distortions, while preserving *local* metric structure – inducing value-based clusters in the state space. This is in contrast to existing approaches that attempt to induce *globally* isometric representations. Our proposed representations, which we call Behavrioal Eigenmaps (BeigeMaps), correspond to neural eigenfunctions of similarity kernels induced by bisimulation metrics. Secondly, we also introduce a framework for metric-aware policy composition to generate composite-policies in multi-objective RL as centroids in distance spaces where policies are embedded. These policy centroids not only encapsulate some existing RL policy compositions, but also extend them to new problem settings. In summary, we tackle the following research questions in Part 4 of this dissertation:

Part 4: Metric Informed Reinforcement Learning
[Chapter 9] How can we utilize *bisimulation metrics* in *behavioral representation learning* algorithms in RL to improve *policy performance*.
[Chapter 10] How can we utilize *policy-induced distance spaces* for *policy composition* in multi-objective RL to induce *desirable composition properties*.

1.2 Contributions and Outline

We now provide an outline of the subsequent chapters, and also summarize the contributions we provide in each chapter:

Metric Informed Quantum Graphical Models

1. In Chapter 4, we introduce an approach to learning HQMMs via Riemannian gradient descent, and demonstrate that it enables learning these models with greater accuracy and speed than the prior approach (Srinivasan et al., 2018c).
2. In Chapter 5, we introduce Liouvile-HQMMs, a reformulation of the standard HQMMs that demonstrates that HQMMs are a special class of observable operator models (OOMs), and are more expressive[2] than both HMMs and norm-observable operator models (NOOMs) while still avoiding the negative probability problem. To facilitate this proof, we first prove a different relationship of independent interest: NOOMs do not contain HMMs in terms of expressiveness, i.e., HMM $\not\subseteq$ NOOMs.
3. In Chapter 6, we introduce Input-Output HQMMs that append standard HQMMs with control, and subsume quantum observable Markov decision processes (QOMDPs) (Barry et al., 2014) and quantum Markov decision processes (QuaMDPs) (Cidre, 2016), quantum-inspired generalizations of partially observable Markov decision processes (POMDPs). Using this hierarchy of expressiveness, we show that perfect goal state reachability is undecidable for IO-OOMs, by generalizing known results for QOMDPs.
4. In Chapter 7, we show that uniform matrix product states (uMPS) are equivalent to OOMs (or PSRs or SWA) in the *non-terminating* limit of infinitely long sequences. We show that a specific instance of uMPS known as uniform Born machine (uBMs), are equivalent to NOOMs (Zhao and Jaeger, 2010b) or quadratic weighted automata (Bailly, 2011), and cannot model all finite dimensional HMMs or probabilistic automata. Finally, we demonstrate that another sub-class of uMPS, called uniform locally purified states

[2] We define a model class as more expressive than another if a finite-dimensional model in the former class cannot be exactly modeled by a finite-dimensional model in the latter class. We provide a formal definition in Chapter 3

(uLPS) (Glasser et al., 2019) are equivalent to HQMMs, and thus subsume HMMs, NOOMs, and uBMs.

The contributions presented in Chapters 4-7 were previously published as part of the following publications:

- **Adhikary, S.**, Srinivasan, S., Gordon, G., and Boots., B. (2020). *Expressiveness and Learning of Hidden Quantum Markov Models*. In Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics, PMLR.
- **Adhikary, S.**, Srinivasan, S., Miller, J., Rabusseau, G., and Boots, B. (2021). *Quantum tensor networks, Stochastic Processes, and Weighted Automata*. In Proceedings of the 24th International Conference on Artificial Intelligence and Statistics, PMLR.

Metric Informed Sampling

1. In Chapter 8, we introduce an approach to adapt the kernel herding algorithm (Chen et al., 2010) to sampling over Riemannian manifolds using (a) geometry-aware kernels that incorporate the manifold-specific distances, and (b) Riemannian optimization to constrain generated samples to lie on the manifold.
2. We empirically evaluate our approach on two tasks involving sampling from distributions defined on Riemannian manifolds. First, we demonstrate improved convergence when drawing resamples from manifold-supported distributions, outperforming herding with heuristic projections and importance resampling with optimal transport maps (Wang and Solo, 2020). Second, we apply our method to kernel recursive approximate Bayesian computation to estimate simulator model parameters on the symmetric positive-definite manifold, showing improved accuracy over both heuristic projections and the widely used Cholesky parameterization.

The contributions presented in Chapter 8 were previously published as part of the following publications:

- **Adhikary, S.** and Boots, B. (2022). *Sampling over Riemannian Manifolds using Kernel Herding*. IEEE International Conference on Robotics and Automation (ICRA)

Metric Informed Reinforcement Learning

1. In Chapter 9, we present Behavioral Eigenmaps (BeigeMaps) as RL state representations corresponding to neural eigendecompositions of similarity kernels with respect to bisimulation metrics. We demonstrate that by focusing on preserving the local metric structure induced by bisimulation metrics, BeigeMaps improve the policy performance of prior behavioral distance based RL algorithms as evaluated on the Deep Mind control suite (Tassa et al., 2018) with image observations.
2. In Chapter 10, we present a new framework for policy composition in multi-objective reinforcement learning, viewing the problem as computing centroids in distance spaces where policies are embedded. We demonstrate through empirical evaluations that policy centroids not only subsume various existing composition techniques, but also allow us to define new policy composition classes with desirable properties.

The contributions presented in Chapter 9 and Chapter 10 were previously published as part of the following publications:

- **Adhikary, S.**, Li. A, and Boots, B. (2024). *BeigeMaps: Behavioral Eigenmaps for Reinforcement Learning from Images*. International Conference on Machine Learning (ICML).
- **Adhikary, S.** and Boots, B. (2022). *Modular Policy Composition with Policy Centroids*. Multidisciplinary Conference on Reinforcement Learning and Decision Making (RLDM)

In addition to the papers highlighted above, the work done as part of this dissertation has also contributed towards the following publications:

1. **Adhikary, S**., Thompson, J., and Boots, B., (2021). *Sampling over Riemannian Manifolds with Kernel Herding*. Robotics: Science and Systems (R:SS 2021) Workshop on Geometry and Topology in Robotics
2. Srinivasan, S., **Adhikary S.**, Miller, J., Pokharel, B., Rabusseau, G. and Boots, B., (2021). *Towards a Trace-Preserving Tensor Network Representation of Quantum Channels*. Second Workshop on Quantum Tensor Networks in Machine Learning at NeurIPS.
3. Hogan, J., Arenson, M. D., **Adhikary, S.**, Li, K., Zhang, X., Zhang, R., Valdez, J. N., Lynch, R. J., Sun, J., Adams, A. B., &

Patzer, R. E. (2019). *Assessing Predictors of Early and Late Hospital Readmission After Kidney Transplantation*. Transplantation Direct 5(8)

4. Valdez J.N., Fu T., Hogan J., **Adhikary, S.**, Arenson M., Zhang R., Zhang X., Li K., Adams A.B., Patzer R.E., Sun J. (2019), *Interpretable Post-Transplant Graft Failure Prediction Using Decision Tree Neural Networks*. American Journal of Transplantation 19

5. Hogan J., Arenson M., **Adhikary S.**, Li K., Zhang N., Zhang R., Valdez J., Sun J., Adams A.B., Patzer R.E. *Timing matters: Improving Prediction of Hospital Readmission Post Kidney Transplantation*, American Journal of Transplantation 19.

2
Metrics: Preliminaries

2.1 Metrics

A metric is simply way of describing a notion of similarity – or more precisely dissimilarity – between objects. Often these objects are data points, and we may wish to use metrics to classify or cluster a dataset. For instance, in clustering, the choice of metric determines which data points are grouped together, while in generative models, metrics can influence how closely synthesized data resembles real samples. In other instances, these objects may be parameters of models, and we may be interested in quantifying the difference between two competing models. For instance, parameters of structured probabilistic graphical models tend to satisfy geometric constraints such as orthogonality to ensure feasibility of probabilistic operations. Comparing such models requires metrics that capture these geometric constraints.

Regardless of what is being compared, the choice of the metric induces specific geometric structure on the space of objects, which can be a powerful way to incorporate useful structural priors in algorithms. In some instances, we may have prior domain knowledge about the geometric structure of a problem, and can induce this structure in an algorithm by using the appropriate metric. In other instances, the prior may come in the form of the metric itself – we may be provided with distances between objects, and must use it to identify appropriate representations of the underlying object space. Appropriately utilizing such priors can be crucial in ensuring feasibility of learned models, improving predictive accuracy and generalization, and reducing sample complexity when data is scarce or noisy.

Formally, we define a metric as follows:

Definition 1 (Metric)**.** *A metric* $d : \mathcal{M} \times \mathcal{M} \to [0, \infty)$ *defined over a domain* $\mathcal{M}$ *is a map* d *such that for all* $\mathbf{x}, \mathbf{x}', \mathbf{x}'' \in \mathcal{M}$

1. *Reflexiveness:* $\mathbf{x} = \mathbf{x}' \Rightarrow d(\mathbf{x}, \mathbf{x}') = 0$
2. *Non-degeneracy:* $d(\mathbf{x}, \mathbf{x}') = 0 \Rightarrow \mathbf{x} = \mathbf{x}'$
3. *Symmetry:* $d(\mathbf{x}, \mathbf{x}') = d(\mathbf{x}', \mathbf{x})$
4. *Triangle Inequality:* $d(\mathbf{x}, \mathbf{x}'') \leq d(\mathbf{x}, \mathbf{x}') + d(\mathbf{x}', \mathbf{x}'')$

If we relax some of the constraints above, we obtain other pseudo-metrics, which, although not quite true metrics, can still be useful in many applications. For instance, semi-metrics satisfy all but the non-degeneracy property. Further relaxation of the constraints gives us the broader class of distance functions. We define these pseudo-metrics as follows:

Definition 2. ***Semi-Metric*** *A semi-metric* $d : \mathcal{M} \times \mathcal{M} \to [0, \infty)$ *defined over a domain* $\mathcal{M}$ *is a map* d *such that for all* $\mathbf{x}, \mathbf{x}', \mathbf{x}'' \in \mathcal{M}$*, it satisfies the reflexiveness (1), symmetry (3), and the triangle inequality (4) properties in Definition 1.*

Definition 3. ***Distance*** *A distance* $d : \mathcal{M} \times \mathcal{M} \to [0, \infty)$ *defined over a domain* $\mathcal{M}$ *is a map* d *such that for all* $\mathbf{x}, \mathbf{x}' \in \mathcal{M}$*, it satisfies the reflexiveness (1) and symmetry (3) in Definition 1.*

Outline In the following sections, we discuss a few specific metrics and pseudo-metrics to highlight the variety of metric structures encountered in machine learning, and how these structures can be exploited by algorithms. We begin in Section 2.2 with the Riemannian metric, which encapsulates notions of curvature crucial to formalizing non-Euclidean curved spaces. In Section 2.3 we discuss probability metrics that allow incorporating metrics over a space into metrics comparing distributions defined over the same space. Finally, in Section 2.4, we discuss the bisimulation metric which defines a notion of behavioral similarity used to learn representations in reinforcement learning. While the limited metrics discussed below only provide a small sample of the rich variety of metrics used in machine learning, they illustrate the diversity and utility of metrics. Moreover, we utilize these metrics throughout this dissertation and, as such, the following sections also provide background for subsequent chapters.

2.2 *Riemannian Metrics*

The simplest geometric structure encountered in most applications is that of a flat Euclidean space. Here, the Euclidean metric is simply the length of the straight line connecting any two points. However, we often encounter data and parameter spaces that are better modeled as points on curved non-Euclidean spaces. We can generalize

Euclidean spaces to account for such curvature via *manifolds*, which we now describe.

Manifolds An n-dimensional manifold $\mathcal{M}$ is a topological[1] space that is locally homeomorphic to $\mathbb{R}^n$. Intuitively, a manifold is a space that is similar to Euclidean space within a small neighborhood around any point . If the manifold is additionally differentiable, we can define *tangent spaces* along it. For a point $\mathbf{x}$ on a differentiable manifold, the vector space spanned by the tangent vectors of all possible curves passing through $\mathbf{x}$ is called the tangent space $\mathcal{T}_{\mathbf{x}}\mathcal{M}$. Essentially, this tangent space is one way of defining a Euclidean space that the manifold is similar to within a small neighborhood around $\mathbf{x}$. Note that the tangent space at $\mathbf{x}$ is, in general, different from the tangent space defined at any other point. The Euclidean manifold is a special case where all tangent spaces are equivalent.

[1] Intuitively, a topological space is identified by a notion of a neighborhood around points. A metric space uses a specific notion of neighborhoods defined by a metric function.

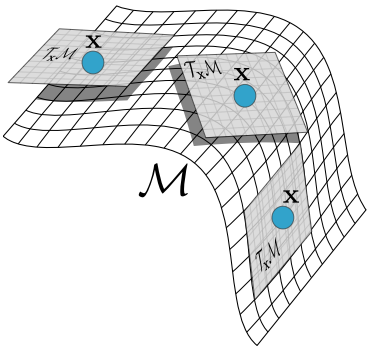


Figure 2.1: **Illustration of Tangent Spaces of Riemannian Manifolds**: A Riemannian manifold $\mathcal{M}$ is homeomorphic to a Euclidean tangent space $\mathcal{T}_{\mathbf{x}}\mathcal{M}$ at every point $\mathbf{x} \in \mathcal{M}$.

The Metric Tensor A metric tensor $g_{\mathbf{x}}$ at a point $\mathbf{x} \in \mathcal{M}$ is a biliear form defined on the tangent space at $\mathbf{x}$ that maps pairs of tangent vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathcal{T}_{\mathbf{x}}\mathcal{M}$ to real numbers, i.e., $g_{\mathbf{x}} : \mathcal{T}_{\mathbf{x}}\mathcal{M} \times \mathcal{T}_{\mathbf{x}}\mathcal{M} \to \mathbb{R}$. Given a local coordinate system $\mathbf{x} = (x^1, \cdots, x^k)$, we obtain a basis $\frac{\partial}{\partial \mathbf{x}} = (\partial/\partial x^1), \cdots, \partial/\partial x^k) = (\partial_1, \cdots, \partial_k)$ on the tangent spaces of the manifold $\mathcal{M}$. The metric tensor $g_{\mathbf{x}}$ can then be expressed in this basis as the *local representation* of the metric – a positive definite matrix $\mathbf{G}(\mathbf{x}) = [g_{ij}(\mathbf{x})]$, where $g_{ij}(\mathbf{x}) = \langle \partial_i | \partial_j \rangle$. This local representation defines an inner product on the tangent space $\mathcal{T}_{\mathbf{x}}\mathcal{M}$.

$$\langle \boldsymbol{v} | \boldsymbol{w} \rangle_{\mathbf{G}(\mathbf{x})} = \boldsymbol{v}^T \mathbf{G}(\mathbf{x}) \boldsymbol{w} \quad \forall \boldsymbol{v}, \boldsymbol{w} \in \mathcal{T}_{\mathbf{x}}\mathcal{M}$$

Riemannian Manifolds A *Riemannian manifold* is a differentiable manifold that exhibits some additional notions of smoothness. Specifically, it is a differentiable manifold with a family of smoothly varying inner products defined on its tangent spaces. Many data and parameter spaces commonly encountered in machine learning can be modeled as Riemannian manifolds. For instance, unit quaternions used to model orientation in vision and robotics can be modeled as L_2-normalized vectors of the S^3 manifold. The general S^n manifold defines an n-dimensional hypershpere, which for $n = 2$ can be visualized as a familiar 3D-sphere as shown in Figure 2.2. Moreover, the operators that induce rotations that alter orientations can also be defined as 3×3 rotation matrices on the $SO(3)$ manifold. The SPD manifold of symmetric-positive definite matrices is also routinely encountered when modeling physical properties such as inertia or stiffness. In 3-dimensions, the SPD manifold carves out a conical structure as shown in Figure 2.3

Geodesic Curves On a flat Euclidean manifold, the shortest path between any two points is just the length of the straight line connecting them. But for non-Euclidean manifolds, the path between two points is a *curve* γ along the space. The length of such curves is obtained by integrating its time-derivatives $\dot{\gamma}(t)$ along it, where t tracks positions along the curve. Specifically, we have

$$\texttt{len}(\gamma) = \int_0^1 \sqrt{\langle \dot{\gamma}(t), \dot{\gamma}(t) \rangle}_{\mathbf{G}(\mathbf{x}_t)} dt$$

Note that if $\mathcal{M}$ is a Riemannian manifold, we know that there exists an inner-product $\langle \cdot, \cdot \rangle_{\mathbf{G}}$ that varies smoothly along curves γ.

A *geodesic curve* on $\mathcal{M}$ is one that locally minimizes distances along it. For any two points on a geodesic curve that are within some ϵ-ball of one another, following the curve corresponds to the shortest path between them. However, the geodesic curve may not be the shortest path[2] between points on the curve that are far apart; the geodesic curve only needs to minimize distances *locally*.

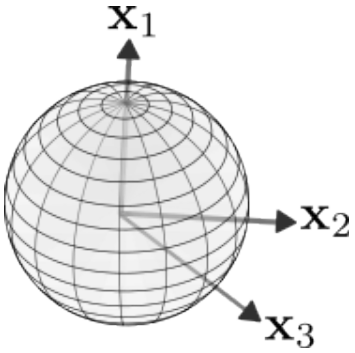


Figure 2.2: **The S^2 manifold**: The hyperspherical manifold S^n consists of $n+1$-dimensional L_2-normalized vectors.

Geodesic Distance Given that there can be multiple geodesic curves connecting any two points on $\mathcal{M}$, the natural choice to define a notion of distance on the manifold is the length of the shortest of these curves. This is defined as the *geodesic distance* of the manifold

$$\text{Geodesic Distance: } d(\mathbf{x}, \mathbf{y}) = \min_{\gamma} \texttt{len}(\gamma) \;\; s.t. \;\; \gamma(0) = \mathbf{x}, \gamma(1) = \mathbf{y}$$

We can derive closed form formulae for geodesic distances on many common Riemannian manifolds.

[2] For instance, we can move from one point to another on a circle either in clockwise or anticlockwise direction. Both of these paths trace out geodesic curves, and one can be shorter than the other.

A Riemannian metric of manifold $\mathcal{M}$ corresponds to the geodesic distance defined with respect to a Riemannian metric tensor $\mathbf{G}$ which defines a smoothly varying inner-product $\langle \cdot, \cdot \rangle_{\mathbf{G}}$.

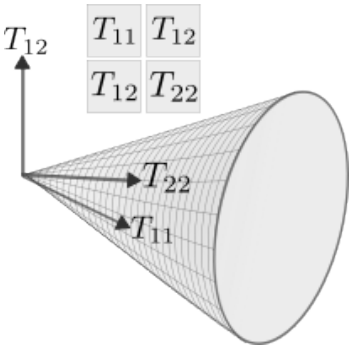


Figure 2.3: **The SPD manifold**: Illustration of the SPD manifold of 3×3 matrices with elements $[[T_{11}, T_{12}], [T_{12}, T_{22}]]$.

Optimization over Riemannian Manifolds Given data that lie on Riemannian manifolds, a naive approach is to simply ignore the curvature of the space and approximate it as flat Euclidean space. By the definition of Riemannian manifolds, this approximation breaks down beyond a neighborhood around any point in the space. Algorithms that do not take this assumption into account can lead to infeasible or inaccurate results. This limitation is particularly evident for optimization or search algorithms – ignoring the curvature of the underlying space can lead an optimization routine to land on solutions that do not satisfy the constraints of the problem. Equipped with notions of curves, distances, and tangent spaces on Riemannian manifolds, we

can generalize many Euclidean optimization algorithms to Riemannian manifolds (Absil et al., 2007b; Smith, 1994). In the subsequent chapters, we will particularly rely on optimizations based on gradient descent, which we now discuss in detail.

Gradient descent on Euclidean manifolds follows a simple recipe. We take a step in the direction opposite to the gradient of an objective function with respect to a parameter by subtracting the gradient from it. Viewing the Euclidean manifold as a Riemannian manifold, the gradient $\boldsymbol{g}(\mathbf{x})$ of the objective function at a point $\mathbf{x}$ is an element of the tangent space $\mathcal{T}_{\mathbf{x}}\mathcal{M}$. Subtracting $\boldsymbol{g}(\mathbf{x})$ from $\mathbf{x}$ only makes sense because the tangent spaces of Euclidean manifolds are also Euclidean. To perform gradient descent on general Riemannian manifolds, we need to first map gradients from tangent spaces onto the manifold using an *exponential map* $\mathrm{Exp}_{\mathbf{x}}\mathcal{M} : \mathcal{T}_{\mathbf{x}}\mathcal{M} \to \mathcal{M}$. For any point $\mathbf{x} \in \mathcal{M}$, and a tangent vector $\boldsymbol{g}(\mathbf{x}) \in \mathcal{T}_{\mathbf{x}}\mathcal{M}$, there exists a unique geodesic curve $\tilde{\gamma}(t)$ such that $\tilde{\gamma}(0) = \mathbf{x}$ and $\frac{d}{dt}\tilde{\gamma}(t)|_{t=0} = \boldsymbol{g}(\mathbf{x})$. The exponential map $\mathrm{Exp}_{\mathbf{x}}\mathcal{M}$ applied to $\alpha\boldsymbol{g}(\mathbf{x})$ corresponds to motion along the geodesic $\tilde{\gamma}$ for $\alpha \in \mathbb{R}_{\geq 0}$ steps

$$\mathrm{Exp}_{\mathbf{x}}\mathcal{M}\left(\alpha\boldsymbol{g}(\mathbf{x})\right) = \tilde{\gamma}(\alpha)$$

Intuitively, the exponential map defines the curve on the manifold along which we would move if we were to start at $\mathbf{x}$ with velocity $\alpha\boldsymbol{g}(\mathbf{x})$ and zero acceleration. This exponential map applied to the negative gradient of the objective function corresponds to gradient descent with step-size α. The inverse of the exponential map, called the *logarithmic map* $\mathrm{Log}_{\mathbf{x}}\mathcal{M}$, defines the reverse mapping $\mathcal{M} \to \mathcal{T}_{\mathbf{x}}\mathcal{M}$. We provide an illustration of using exponential maps for gradient descent in Figure 2.4.

As standard gradient descent can be noisy and slow to converge, it is generally supplemented with additional optimization techniques such as the conjugate gradient method (Hestenes and Stiefel, 1952; Dai and Yuan, 1999), momentum (Polyak, 1964; Qian, 1999a), RMSprop, Adam (Kingma and Ba, 2015), etc. These techniques combine the current gradient of the objective with gradients computed at earlier iterations to form of a new descent direction. Note that this strategy implicitly assumes that gradients computed at different points are elements of the same vector space, and can thus be combined. This is true for Euclidean manifolds, but not for Riemannian manifolds in general. To combine gradients computed at different points on a Riemannian manifold, we can use the *parallel transport map* $\Gamma_{\mathbf{x}\to\mathbf{y}} : \mathcal{T}_{\mathbf{x}}\mathcal{M} \to \mathcal{T}_{\mathbf{y}}\mathcal{M}$, which maps elements from one tangent space to another while preserving inner products. Intuitively, a parallel transport map is a change of reference frame between tangent spaces that preserves relative distances.

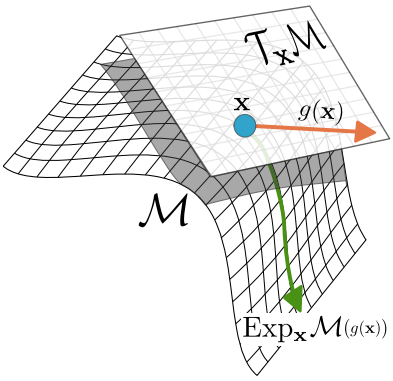


Figure 2.4: **Illustration of Gradient Descent over Riemannian Manifolds**: The gradient $g(\mathbf{x})$ at a point $\mathbf{x}$ on a Riemannian manifold $\mathcal{M}$ is a vector on the tangent space $\mathcal{T}_{\mathbf{x}}\mathcal{M}$. Stepping along $\boldsymbol{g}(\mathbf{x})$ would, in general, result in going off the manifold, whereas stepping along $\mathrm{Exp}_{\mathbf{x}}\mathcal{M}(\boldsymbol{g}(\mathbf{x})$ follows a geodesic on the manifold.

Applications of Riemannian Manifolds Riemannian manifolds have found widespread use in machine learning due to their ability to model non-Euclidean structure in data and parameter spaces that arise naturally in many learning problems. Applications span diverse areas including optimization (Boumal, 2023; Absil et al., 2007b; Smith, 1994; Kochurov et al., 2020; Dreisigmeyer, 2007), representation learning (Chen et al., 2023; Nickel and Kiela, 2017; Ganea et al., 2018), statistical analysis (Pennec, 2006; Fletcher et al., 2004; Sommer et al., 2010; Calinon, 2020), natural language processing (Reisinger et al., 2010; Tifrea et al., 2018), protein structure modeling (Mardia et al., 2007), bayesian optimization (Jaquier et al., 2020), robotics and control (Braun et al., 2024; Zeestraten et al., 2017; Cheng et al., 2021; Li et al., 2021) and many more.

In Chapter 4 we use Riemannian gradient descent to learn probabilistic graphical models known as hidden quantum Markov models (HQMMs), the parameters of which lie on a specific Riemannian manifold known as the Stiefel manifold. In Chapter 8, we apply Riemannian gradient descent over a wider range of Riemannian manifolds as part of the kernel herding algorithm to draw samples from distributions. In addition to Riemannian optimization, we also make use of the geodesic distances for these manifolds to formulate geometry-aware similarity metrics that inform the quality of the sampling protocol.

2.3 *Probability Metrics*

Metrics can be defined not only over data spaces but also over the space of probability distributions defined on those spaces. These probability metrics quantify differences between distributions and play a central role in many areas of machine learning, especially in optimization and inference. Divergences and distances between probability distributions are broadly applicable in machine learning. Divergences such as the Kullback-Leibler (KL) divergence, total variation distance, Rényi divergence, Hellinger distance, and chi-squared divergence are widely used in machine learning. These measures provide different ways to quantify dissimilarity between probability distributions, with each offering distinct advantages depending on the structure and requirements of the problem.

In the following sections, we focus on two widely studied probability metrics which allow us to incorporate a ground metric defined on a space into the distance between distributions defined over the same space. We also use these two distances in the subsequent chapters: the Maximum Mean Discrepancy (MMD) (Gretton et al., 2012) and Wasserstein distances (Peyré et al., 2019).

2.3.1 *Maximum Mean Discrepancy*

Kernels In most machine learning applications, data is mapped from its domain space $\mathcal{X}$ onto a feature space Φ via a potentially non-linear feature map $\phi : \mathcal{X} \to \Phi$. In many of these applications, the actual features themselves, say $\phi(\mathbf{x})$ and $\phi(\mathbf{y})$, are not utilized directly; instead, the relative similarities between the features are of interest. Such similarities are generally computed via an inner-product, also known as the *kernel*: $k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle_\Phi = \phi(\mathbf{x})^\dagger \phi(\mathbf{y}) \in \mathbb{R}_{\geq 0}$. Even though the kernel is equivalent to the inner-product, it may be possible to evaluate it directly without having to explicitly calculate the inner-product, which would avoid using, or even storing, the potentially high-dimensional vectors $\phi(\mathbf{x})$ and $\phi(\mathbf{y})$; this is the popular 'kernel trick'.

In addition to being implicitly defined with respect to a function ϕ over $\mathcal{X}$, the kernel itself can be viewed as a function over $\mathcal{X}$. Namely, given a data point $\mathbf{x}$, the function $k(\mathbf{x}, \cdot) : \mathcal{X} \to \mathbb{R}_{\geq 0}$ maps datapoints to non-negative reals, and the kernel itself corresponds to an infinite-dimensional feature 'vector' $\boldsymbol{k}$.

Reproducing Kernel Hilbert Spaces Instead of inferring the kernel from a feature map, it is generally more convenient to define the kernel directly. If a kernel is symmetric and positive definite such that $\sum_{ij} c_i c_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0$ for all $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$ and $c_i, c_j \in \mathbb{R}$, the function $k(\mathbf{x}, \mathbf{x}')$ is the inner product of a feature map from $\mathcal{X}$ to $\mathcal{H}$, a potentially infinite-dimensional Hilbert space (Aronszajn, 1950; Muandet et al., 2017). In fact, the function $k(\mathbf{x}, \cdot) \in \mathcal{H}$ itself defines the *canonical* features of $\mathcal{X}$ in $\mathcal{H}$, and the kernel is the canonical feature map i.e., $k : \mathcal{X} \to \mathcal{H}$.

The Hilbert space $\mathcal{H}$ associated with SPD kernels has some additional structure; it is actually a *reproducing kernel* Hilbert space (RKHS) (Aronszajn, 1950) that satisfies the following reproducing property

$$f(\mathbf{x}) = \langle k(\mathbf{x}, \cdot), \mathbf{f} \rangle_{\mathcal{H}} \quad \forall\ \mathbf{f} \in \mathcal{H} \ \text{ and } \ \mathbf{x} \in \mathcal{X}$$

We can thus evaluate non-linear functions f representable in the RKHS through a linear inner product. Moreover, all functions that can be represented in an RKHS $\mathcal{H}$ are bounded such that $|f(\mathbf{x})| \leq C||\mathbf{f}||_{\mathcal{H}}$ for some $C > 0$. This kernel-specific constant C imposes an inherent smoothness regularization when computing solutions in $\mathcal{H}$.

Kernel Mean Embeddings Besides embedding individual samples from $\mathcal{X}$ to an RKHS, we can also embed entire distributions defined over $\mathcal{X}$. A probability distribution P is embedded in an RKHS as a

kernel mean embedding $\boldsymbol{\mu}_P$, corresponding to the expected value of the kernel function $\mathbb{E}_{\mathbf{x}\sim P} k(\mathbf{x}, \cdot)$. We define $\boldsymbol{\mu}_P$ and its empirical estimate $\hat{\boldsymbol{\mu}}_P$ as follows

$$\boldsymbol{\mu}_P = \int k(\mathbf{x}, \cdot) dP(\mathbf{x}) \qquad \hat{\boldsymbol{\mu}}_P = \frac{1}{n} \sum_{i=1}^{n} k(\mathbf{x}_i, \cdot) \;\text{ for } \mathbf{x}_i \sim P$$

The empirical estimate $\hat{\boldsymbol{\mu}}_P$ converges to $\boldsymbol{\mu}_P$ at a rate of $O(n^{-1/2})$ (Smola et al., 2007).

We can also construct the empirical mean embedding as a *weighted* average $\hat{\boldsymbol{\mu}}_P = \sum_{i=1}^{n} w_i k(\mathbf{x}_i, \cdot)$ from points $\{\mathbf{x}_i\}$ not necessarily sampled from P. Note that the weights w_i can, in general, be negative; preventing us from using various off-the-shelf sampling techniques. Given a kernel mean embedding of a distribution, we can calculate expected values of functions $\mathbf{f} \in \mathcal{H}$ as follows

$$\mathbb{E}_{\mathbf{x}\sim P}(f(x)) = \langle \mathbf{f},\ \boldsymbol{\mu}_P \rangle_{\mathcal{H}} \approx \langle \mathbf{f}, \hat{\boldsymbol{\mu}}_P \rangle$$

If the kernel is additionally *characteristic*, the mapping from P to $\boldsymbol{\mu}_P$ is injective, and $\boldsymbol{\mu}_P$ is the complete, unique description of P (Fukumizu et al., 2007). Popular examples of characteristic kernels over Euclidean space include the Gaussian and Laplacian kernels (Fukumizu et al., 2007).

Maximum Mean Discrepancy The kernel mean embeddings of distributions can be used to define a metric for probability distributions. Essentially, kernel mean embeddings map distributions onto points on the RKHS $\mathcal{H}$, and the distance between two distributions can be defined essentially as the distance between points in this space.

$$MMD_{\mathcal{H}_k}[P, Q] = \|\boldsymbol{\mu}_P - \boldsymbol{\mu}_Q\|_{\mathcal{H}_k} \approx \widehat{MMD}_{\mathcal{H}_k} = \|\hat{\boldsymbol{\mu}}_P - \hat{\boldsymbol{\mu}}_Q\|_{\mathcal{H}_k},$$

where $\widehat{MMD}$ corresponds to the empirical estimate of the MMD[3].

In general, the MMD is a semi-metric[4] as two non-identical distributions can be mapped to the same kernel mean embeddings. However, if the kernel k is characteristic, the MMD is a proper metric, i.e., $MMD[P, Q] = 0 \Leftrightarrow P = Q$.

Applications of the MDD The maximum mean discrepancy has found a wide range of applications, especially in problems that require comparing distributions without explicit density estimation. Some exmaples include two-sample testing (Gretton et al., 2012), domain adaptation (Quiñonero-Candela et al., 2008) generative modeling (Li et al., 2017a), causal inference (Shalit et al., 2017), domain generalization (Muandet et al., 2013), and so on.

[3] We can also obtain an unbiased estimate of the MMD using kernel evaluations directly without first estimating the kernel mean embeddings (Muandet et al., 2017; Borgwardt and Patterson, 2021).

[4] The MMD is actually part of a broader class of semi-metrics known as integral probability metrics (IPM) defined as follows:

$$\sup_{f\in\mathcal{F}} \left[\mathbb{E}_{\mathbf{x}\sim P}[f(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim Q}[f(\mathbf{x})]\right]$$

Different IPMs can be constructured with specific choice of the domain $\mathcal{F}$. In the case of the MMD, $\mathcal{F}$ is set to be the unit-norm ball of the RKHS $\mathcal{H}_k$ induced by the kernel k (Muandet et al., 2017).

In Chapter 8, we present a Riemannian adaptation of the kernel herding algorithm (Chen et al., 2010), which draws samples from kernel mean embeddings of distributions. These samples minimize the MDD between the target distribution and the empirical distribution of drawn samples. In Chapter 10, we define MMD-centroids to aggregate policies in multi-objective decision making (Roijers et al., 2013) problems through actions sampled from individual policies.

2.3.2 *Wasserstein Distances*

Optimal Transport Given a metric space $(\mathcal{M}, d)$, the optimal transport problem is concerned with transporting a distribution P defined on $\mathcal{M}$ to another distribution Q defined on $\mathcal{M}$. Intuitively, a distribution P can be thought of as assigning some probability mass to points in $\mathcal{M}$, and a transportation plan moves probability mass to other points to match the distribution Q. The optimal transport plan is one that achieves this matching with minimal effort, where effort is measured with respect to the distance $d(\mathbf{x}, T(\mathbf{x}))$ that the original particles $\mathbf{x}$ must be moved.

Formally, given a map $T : \mathcal{M} \to \mathcal{M}$, we can define a *pushforward-map* $T_{\#}P$ as follows:

$$T_{\#}P(\mathbf{x}) = P(T^{-1}(\mathbf{x}))$$

Essentially, the push-forward $T_{\#}$ assigns the probability $P(\mathbf{x})$ to the transported point $T(\mathbf{x})$. We specifically seek push-forwards that result in $T_{\#}P = Q$, and even within that set, we want to find a specific push-forward that minimizes the total cost of moving the particles with respect to the ground-metric d. This induces the *optimal transport distance* between distributions defined as the solution to the following Monge problem (Monge, 1781; Peyré et al., 2019):

$$\inf_{T:T_{\#}=Q} \int d(\mathbf{x}, T(\mathbf{x}))^p dP(\mathbf{x})$$

However, the minimizer T^* of the above objective does not always exist[5] – there may just not be a transport map such that $T_{\#}P = Q$. To account for this, the above optimal transport problem can be relaxed into the Kantorovich formulation, which essentially allows the probability mass at points $\mathbf{x}$ to be split into multiple locations (Kantorovich, 2006; Peyré et al., 2019). This relaxed Kantorovich distance, commonly known as *Wasserstein distance* W_p, is defined as follows:

$$W_p(P, Q) = \left(\inf_{J \in \mathcal{J}(P,Q)} \int_{\mathcal{M}^2} d(\mathbf{x}, \mathbf{y})^p dJ(\mathbf{x}, \mathbf{y}) \right)^{1/p}$$

where $\mathcal{J}(P, Q)$ is the set of all joint distributions of P and Q with marginals P and Q. Different values of the subscript p correspond to

[5] For instance, consider the case where P is a dirac distribution centered on a single point $\mathbf{x}$, and Q assigns equal probability of 0.5 to two different points $\mathbf{y}$ and $\mathbf{z}$. The best we can do is move the probability mass at $\mathbf{x}$ to either $\mathbf{y}$ or $\mathbf{z}$, but we can never fully recover Q.

different Wasserstein distances; when $p = 1$, the Wasserstein distance is also known as the earth mover's distance.

Empirical Measures Let us now consider the case where instead of the distributions P and Q, we only have access to their empirical estimates $\hat{P}$ and $\hat{Q}$ defined over samples $\{\mathbf{x}_i \in \mathcal{M}\}$ and $\{\mathbf{y}_i \in \mathcal{M}\}$ defined as

$$\hat{P} = \sum_i^n \alpha_i \mathbf{x}_i \quad \hat{Q} = \sum_j^m \beta_j \mathbf{y}_j \quad \text{s.t.} \quad \sum_i \alpha_i = \|\boldsymbol{\alpha}\| = 1 \;\&\; \sum_j \beta_j = \|\boldsymbol{\beta}\| = 1,$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are normalized weight vectors.

In this setting, the empirical joint distribution J as well as pairwise distances $d^p(\mathbf{x}, \mathbf{y})$ are encoded in $n \times m$ matrices $\mathbf{J}$ and $\mathbf{D}$ respectively. Computing the Wasserstein distance then corresponds to a minimization over transportation plans $\boldsymbol{\gamma}$, also an $n \times m$ matrix, as follows

$$\min_{\boldsymbol{\gamma}} \sum_{i,j} \boldsymbol{\gamma}_{i,j} \mathbf{D}_{i,j} \quad \text{s.t.} \quad \boldsymbol{\gamma}\, \vec{\mathbb{1}}_m = \boldsymbol{\alpha} \quad \boldsymbol{\gamma}^T \vec{\mathbb{1}}_n = \boldsymbol{\beta} \quad \boldsymbol{\gamma} \geq 0$$
$$\text{where} \quad \mathbf{D}_{i,j} = d(\mathbf{x}_i, \mathbf{y}_j)^p \tag{2.1}$$

Essentially, the problem is to find a way to move the total probability mass to define a joint distribution $J \in \mathcal{J}(P, Q)$ with marginals P and Q that minimizes the total weighted distance. Formally, this corresponds to finding the p-th root of the optimum network flow problem (Cuturi and Doucet, 2014; Peyré et al., 2019).

Calculating Wasserstein distances is in general computationally expensive – it scales at least in $O(n^3 \log n)$ for empirical distributions with n samples. In practice, the objective is generally relaxed with an entropic regularization term, resulting in a family of Sinkhorn distances[6] that can be optimized much faster (Cuturi, 2013).

[6] Feydy et al. (2019) show that the family of Sinkhorn divergences can be used to interpolate between MMD and Wasserstein distances.

Applications of Wasserstein Metrics Wasserstein distances have served as a powerful tool in machine learning to compare probability distributions in a geometry-aware manner. It has been widely applied in generative modeling (Arjovsky et al., 2017; Gulrajani et al., 2017; Deshpande et al., 2018) , clustering (Ho et al., 2017), domain adaptation (Courty et al., 2016; Montesuma and Mboula, 2021), variational inference (Lambert et al., 2022; Tolstikhin et al., 2017), graphics and shape analysis (Bonneel and Digne, 2023), and so on.

In Chapter 8, we use the Wasserstein metric to compare the quality of samples drawn from target distributions defined over Riemannian manifolds. In Chapter 9, we use another metric known as the bisimulation metric for reinforcement learning, which is defined with respect to the Wasserstein distance. We now discuss this metric in detail.

2.4 *Bisimulation Metrics*

In prior sections, we have presented various instances where the metric structure of data or parameters can inform the design of optimization or modeling algorithms. Namely, when data or parameters lie on Riemannian manifolds, the geodesic curves of the manifolds provide paths for curvilinear optimization, and the geodesic distances provide geometry-aware notions of similarities. Moreover, such metrics defined on a space can be embedded into metrics over probability distributions. We now turn to a problem setting where a problem-specific metric is not provided a priori, but must be learned from data. These metrics can then be used to learn geometry-aware data representations. Specifically, we discuss behavioral distances used to aggregate states leading to similar outcomes.

2.4.1 *Reinforcement Learning in Markov Decision Processes*

We formalize the problem setting for reinforcement learning as a Markov decision process (MDP), which is defined as a tuple $(\mathcal{X}, \mathcal{A}, r, \mathcal{P}, \gamma)$. Here, $\mathcal{X}$ and $\mathcal{A}$ are continuous state and action spaces with dimensions $\texttt{dim}(\mathcal{X})$ and $\texttt{dim}(\mathcal{A})$, $r : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. The distribution $\mathcal{P}^a_{\mathbf{x}} = \mathcal{P}(\mathbf{x}'|\mathbf{x}, a)$ encodes transition probabilities from $\mathbf{x}$ to $\mathbf{x}'$ under action a. The RL problem corresponds to training a policy $\pi : \mathcal{X} \to \mathcal{A}$ that maximizes the expected discounted return $\mathbb{E}_\pi\big[\sum_0^H \gamma^t r^a_{\mathbf{x}_t}\big]$, where $r^a_{\mathbf{x}} = r(\mathbf{x}, a)$, and H is the (potentially infinite) time horizon. The expectation $\mathbb{E}_\pi$ is taken over $\mathcal{P}^\pi_{\mathbf{x}}$, the transition distribution under the policy π. A policy's *value* $V^\pi(\mathbf{x}) = \mathbb{E}_\pi\big[\sum_{t=0}^H \gamma^t r^a_{\mathbf{x}_t} | \mathbf{x}_0 = \mathbf{x}\big]$ is the expected return of following π when starting at $\mathbf{x}$.

We use the notation $\mathcal{P}^a_{\mathbf{x}} = \mathcal{P}^a(\mathbf{x}) = \mathcal{P}^a(\mathbf{x}'|\mathbf{x})$ and $r^a_{\mathbf{x}} = r^a(\mathbf{x}) = r(\mathbf{x}, a)$ interchangeably. The expected values of these quantities under a policy π are denoted as $r^\pi_{\mathbf{x}} = r^\pi(\mathbf{x}) = \mathbb{E}_{a\sim\pi}\big[r(\mathbf{x}, a)\big]$ and $\mathcal{P}^\pi_{\mathbf{x}} = \mathcal{P}^\pi(\mathbf{x}) = \mathbb{E}_{a\sim\pi}\big[\mathcal{P}^a(\mathbf{x}'|\mathbf{x})\big]$.

Behavioral distances define a notion of state dissimilarity in reinforcement learning that encodes differences in returns under a sequence of actions. They are termed "behavioral" precisely because they capture long-range temporal differences. These distances are generally variations of the bisimulation metric (Ferns et al., 2011, 2004), which enables learning representations that group states by their value based behavior, which can be useful for downstream policy learning.

2.4.2 *Bisimulation Equivalence*

If the state space of an MDP exhibits sufficient symmetry with respect to the value or policy, representations or state *abstractions* can be used to perform state *aggregation*: assigning equivalent states to the same latent state, and effectively reducing the effective size of the MDP's state space. Instead of such hard aggregation, we might also

opt for softer aggregation where states deemed similar are assigned similar representations.

Both state aggregation and abstraction ultimately hinge on defining an appropriate notion of equivalence. In the context of MDPs, *stochastic bisimulation* has been used extensively to assign equivalence between states based on long term behaviors originating from them. Originating in concurrency theory (Milner, 1980; Park, 1981), stochastic bisimulation was introduced as a relationship between two states that have the same probability of transitioning to classes of equivalent states.

Givan et al. (2003) adapted the concept of bisimulation relations to MDPs by incorporating rewards[7]. In the context of MDPs, stochastic bisimulation, or simply bisimulation is defined as follows:

[7] Givan et al. (2003) consider multiple notions of equivalence including action-sequence equivalence (do two states result in the same reward distributions under any fixed action-sequence?), and optimal value equivalence (do two states have the same optimal value?). They ultimately recommend stochastic bisimulation as an appropriate equivalence criteria for MDPs.

Definition 4. ***Bisimulation Relations (Givan et al., 2003)*** *Given an MDP $(\mathcal{X}, \mathcal{A}, r, \mathcal{P}, \gamma)$, an equivalence relation $\mathfrak{R}$ on $\mathcal{X}$ is a stochastic bisimulation relation*[8] *(or simply a bisimulation relation) if for all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$*

$$\mathbf{x}\mathfrak{R}\mathbf{y} \iff \forall a \in \mathcal{A}, \begin{cases} r^a(\mathbf{x}) = r^a(\mathbf{y}) & \textit{Matching reward} \\ \forall \mathcal{C} \in \mathcal{X}/\mathfrak{R},\ \mathcal{P}^a_{\mathbf{x}}(\mathcal{C}) = \mathcal{P}^a_{\mathbf{y}}(\mathcal{C}) & \textit{Matching transitions} \end{cases}$$

where $\mathcal{P}^a_{\mathbf{x}}(\mathcal{C})$ is the probability of transitioning from state $\mathbf{x}$ to any state in the equivalence class $\mathcal{C}$ under action a.

[8] Given an equivalence relation $\mathfrak{R}$ defined over a domain $\mathcal{X}$, the notation $\mathbf{x}\mathfrak{R}\mathbf{y}$ implies that $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ are equivalent with respect to $\mathfrak{R}$. Moreover, $\mathcal{X}/\mathfrak{R}$ indicates a partition of the space $\mathcal{X}$ with respect to $\mathfrak{R}$ such that every $\mathcal{C} \in S/\mathfrak{R}$ is a subset of equivalent states.

Note that one can define multiple bisimulation relations for a given MDP. Indeed, given a bisimulation relation $\mathfrak{R}$, we could define a coarser or finer equivalence classes and obtain a new relation. Generally, we are interested in the *largest* of these bisimulation relations, which we refer to as *bisimilarity*:

Definition 5. ***Bisimilarity $\sim$ (Givan et al., 2003)*** *Given an MDP $(\mathcal{X}, \mathcal{A}, r, \mathcal{P}, \gamma)$, bisimilarity $\sim$ is the largest stochastic bisimulation relation (Definition 4) defined on the MDP. The notation $\mathbf{x} \sim \mathbf{y}$ indicates that $\mathbf{x}$ and $\mathbf{y}$ are equivalent with respect to $\sim$, i.e., they are bisimilar.*

In the sequel, a bisimulation relation will almost always refer to the bisimilarity relation, unless noted otherwise.

Exact Bisimilarity is Brittle For practical purposes, the notion of exact bisimilarity tends to be too brittle or unstable to be useful. Specifically, minor numerical differences in the parameters of two MDPs can lead to very different state partitions with respect to their corresponding bisimilarities. Moreover, state aggregations defined with respect to an exact bisimilarity may be too conservative – two states cannot be aggregated unless their rewards and equivalence class transitions match *exactly*.

To avoid such instabilities and conservatism, it is useful to generalize binary bisimilarities to *metrics* (Ferns et al., 2004, 2012, 2011) that allow us to define state similarities as opposed to exact matches.

2.4.3 *Bisimulation Metrics*

Bisimulation metrics are semi-metrics or pseudo-metrics that essentially provide a notion of soft-equivalence. A semi-metric satisfies all properties of a metric, except that two non-identical states can have 0 distance between them. It is precisely this property that allow us to encode non-trivial equivalence relations through semi-metrics. Namely, every semi-metric induces an equivalence relation defined as follows:

Definition 6. ***Equivalence Relation Induced by Semimetrics*** *Let* $\texttt{met}$ *be the space of all pseudometrics on a domain* $\mathcal{X}$*. Every pseudometric* $d \in \texttt{met}$ *induces an equivalence relation* $x\mathfrak{R}_d y \iff d(\mathbf{x}, \mathbf{y}) = 0$

Ferns et al. (2012) define the bisimulation metric as a semi-metric that induces the bisimilarity equivalence. This metric[9] is defined as the fixed point of a bisimulation operator $F(d) : \texttt{met} \to \texttt{met}$ that acts on a semi-metric and produces another semi-metric, as defined below:

[9] Here, we consider MDPs with potentially infinite state spaces. See Ferns et al. (2004) for the corresponding forumation for finite MDPs.

Theorem 1 (Ferns et al. (2012))**.** *Let* $\texttt{met}$ *be a uniformly bounded set of lower semi-continuous (lsc) semimetrics*[10]*. Given a scalar* $c \in (0,1)$*, let* $\mathfrak{F} : \texttt{met} \to \texttt{met}$ *be the* ***bisimulation operator*** *acting on a semi-metric function d defined as follows*

$$\mathfrak{F}\, d(\mathbf{x}, \mathbf{y}) = \max_{a \in \mathcal{A}} \left(\left| r^a_{\mathbf{x}} - r^a_{\mathbf{x}} \right| + c\mathcal{W}^h_1\left(\mathcal{P}^a_{\mathbf{x}}, \mathcal{P}^a_{\mathbf{y}}\right)\right) \tag{2.2}$$

[10] A function $h : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is *lower semi-continuous* (lsc) if whenever $(\mathbf{x}_n, \mathbf{x}'_n)$ tends to $(\mathbf{x}, \mathbf{x}')$, then $\liminf h(\mathbf{x}_n, \mathbf{x}'_n) \geq h(\mathbf{x}, \mathbf{x}')$.

where $\mathcal{W}^d_1(\mathcal{P}^a_{\mathbf{x}}, \mathcal{P}_{\mathbf{y}})$ *is the Wasserstein distance between probability distributions* $\mathcal{P}^a_{\mathbf{x}}$ *and* $\mathcal{P}_y$*. The operator* $\mathfrak{F}$ *has a unique fixed point* $d_\sim$*, the* ***bisimulation metric****, the induced equivalence of which corresponds to the bisimilarity, i.e.* $\mathbf{x}\mathfrak{R}_{d_\sim}\mathbf{y} \implies \mathbf{x} \sim \mathbf{y}$

Here, the 1-Wasserstein distance $\mathcal{W}^d_1(\mathcal{P}^a_{\mathbf{x}}, \mathcal{P}^a_y)$ *lifts* the ground-distance d between states in $\mathcal{X}$ to a distance between probability distributions over $\mathcal{X}$. We formally define the bisimulation metric as follows

Definition 7 (Ferns et al. (2012))**.** ***Bisimulation Metric*** $d_\sim$ *Given* $c \in (0,1)$*, the bisimulation metric* $d_{\sim,c}$ *for an MDP corresponds to the fixed point of the bisimulation operator defined in Theorem 1. Following convention, we use* $d_\sim$ *(without reference to c) for the common choice of* $c = \gamma$*.*

Aggregating States with Bisimulation Metrics Given a bisimulation metric $d_\sim$, we can define a state aggregation scheme where states that are relatively close together are treated as the same state. The central idea here is that the aggregated MDP should be simpler to solve than its original counterpart. For instance, we could partition an MDP's state space into a fixed number of clusters, and solve the resulting finite MDP with simple RL techniques as a proxy for the original MDP with a potentially high-dimensional state space. Following the formulation in Kemertas and Jepson (2022), we define state-aggregation formally as follows:

Definition 8 (Kemertas and Jepson (2022)). ***(ϵ, d)-aggregate MDPs***[11] *Given an MDP $(\mathcal{X}, \mathcal{A}, r, \mathcal{P}, \gamma)$, a distance function $d : \mathcal{X} \times \mathcal{X} \to [0, \infty)$, and $\epsilon \geq 0$, let $\tilde{\mathcal{X}}$ be an aggregated state space obtained through the mapping ϕ_ϵ such that*

$$\phi_\epsilon(\mathbf{x}) = \phi_\epsilon(\mathbf{y}) \implies d(\mathbf{x}, \mathbf{y}) \leq 2\epsilon$$

Moreover, let $B_{\phi_\epsilon}(\mathbf{x})$ be the set of neighbors of $\mathbf{x}$ in $\tilde{\mathcal{X}}$ such that

$$B_{\phi_\epsilon}(\mathbf{x}) = \{\mathbf{z} \in \mathcal{X} \mid \phi_\epsilon(\mathbf{x}) = \phi_\epsilon(\mathbf{z})\}$$

Then $(\tilde{\mathcal{X}}, \mathcal{A}, \tilde{r}, \tilde{\mathcal{P}}, \gamma)$ is an (ϵ, δ)-aggregated MDP where $\tilde{r}$ and $\tilde{\mathcal{P}}$ are per-partition weighted averages. Namely, given an aribtrary measure ξ that assigns a positive measure $\xi(B_\phi(\mathbf{x})) > 0$ to all partitions $B_\phi(\mathbf{x})$, these quantities are defined as

$$\tilde{r}^a(\phi(\mathbf{x})) = \frac{1}{\xi(B_\phi(\mathbf{x}))} \int_{z \in B_\phi(\mathbf{x})} r^a(\mathbf{x}) d\xi(\mathbf{z})$$

$$\tilde{\mathcal{P}}^a(\phi(\mathbf{x})) = \frac{1}{\xi(B_\phi(\mathbf{x}))} \int_{\mathbf{z} \in B_\phi(\mathbf{x})} \mathcal{P}^a(z) d\xi(\mathbf{z})$$

If we construct an $(\epsilon, d_\sim)$-aggregated MDP with respect to the bisimulation metric $d_\sim$, we can bound the difference between the optimal value of this aggregate MDP and that of the original MDP. Thus, solving the (potentially) simpler aggregated MDP can be an appropriate proxy to solving the original MDP. Formally, this relation is decribed by the following[12] theorem

Theorem 2. ***Bounding Values of $(\epsilon, d_\sim)$-aggregate MDPs (Kemertas and Jepson, 2022; Ferns et al., 2012)*** *Let $(\mathcal{X}, \mathcal{A}, r, \mathcal{P}, \gamma)$ be an MDP with bisimulation metric $d_\sim$ and optimal value function V^*. Then the optimal value $\tilde{V}^*_\phi$ of its $(\epsilon, d_\sim)$-aggregate MDP satisfies the following bound given $c \geq \gamma$*

$$\|V^* - \tilde{V}^*_\phi\|_\infty \leq \frac{2\epsilon}{(1-\gamma)}$$

Thus, the bisimulation metric can essentially be used to transform an MDP into a simpler MDP where the original states are assigned to clusters corresponding to values of the original MDP.

[11] The corresponding definition of ϵ-aggregated MDPs for finite MDPs (Kemertas and Jepson, 2022) may provide more intuition for the general counterpart presented in the main text.

(ϵ, d)-aggregate MDP for finite MDPs Given a finite MDP $(\mathcal{X}, \mathcal{A}, r, \mathcal{P}, \gamma)$ and $\epsilon \geq 0$, let $\tilde{\mathcal{X}} = \{C_i, \cdots, C_k\}$ be a set of K clusters such that

1. Every $x \in \mathcal{X}$ is assigned to one and only one cluster C_i through a clustering function $C(s) : \mathcal{X} \to [1, k]$.
2. All members of a cluster are no more than ϵ distance apart: $\forall x, y \in \mathcal{X}, C(x) = C(y) \implies d(\mathbf{x}, \mathbf{y}) \leq \epsilon$

Then the ϵ-aggregate MDP is defined as the tuple $(\tilde{\mathcal{X}}, \mathcal{A}, \tilde{r}, \tilde{\mathcal{P}}, \gamma)$ where the aggregated rewards and transitions are averaged over clusters, i.e,

1. $\tilde{r}^a(C_i) = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} r^a(\mathbf{x})$
2. $\tilde{\mathcal{P}}^a_{C_i}(C_j) = \frac{1}{|C_i|} \sum_{x \in C_i} \mathcal{P}^a_{\mathbf{x}}(C_j)$

[12] The corresponding theorem for finite MDPs may be beneficial in building intuition for the general counterpart presented in the main text:

Lipschitz Continuity of Optimal Values (Ferns et al., 2012) Given a finite MDP $(\mathcal{X}, \mathcal{A}, r, \mathcal{P}, \gamma)$, a metric-discount factor $c \in (0, 1)$ such that $\gamma \leq c$, and the bisimulation metric $d_\sim$, we have that

$$\left|V^*(\mathbf{x}) - V^*(\mathbf{y})\right| \leq \frac{1}{1-c} d_\sim(\mathbf{x}, \mathbf{y})$$

In other words, V^* is $\frac{1}{1-c}$-Lipschitz continuous with respect to $d_\sim$.

Bisimulation Metrics as Optimal Value Functions While the above properties relate bisimulation metrics to values of a given MDP $(\mathcal{X}, \mathcal{A}, r, \mathcal{P}, \gamma)$, it turns out that the metric itself can actually be interpreted as the value function of a *lifted* MDP (Ferns and Precup, 2014). The state space of this lifted MDP, or the *self-coupled MDP*, is simply the product space $\mathcal{X} \times \mathcal{X}$. The transition dynamics of this self-coupled MDP is defined as a coupling[13] of $\mathcal{P}$ with itself. At state $([\mathbf{x}, \mathbf{y}]) \in \mathcal{X} \times \mathcal{X}$, the transition dynamics for action $a \in \mathcal{A}$ is defined as a joint distribution with marginals $\mathcal{P}^a_{\mathbf{x}}$ and $\mathcal{P}^a_{\mathbf{y}}$. We define self-coupled MDPs formally as follows

[13] A coupling $\lambda(\mathcal{P}_1, \mathcal{P}_2)$ is a joint-distribution with marginals $\mathcal{P}_1$ and $\mathcal{P}_2$, and Λ denotes the set of all such couplings.

Definition 9. ***Self-Coupled MDP (Ferns and Precup, 2014)*** *Let M be an MDP $(\mathcal{X}, \mathcal{A}, r, \mathcal{P}, \gamma)$ such that $r : \mathcal{X} \times \mathcal{A} \to [0, 1]$ and $(\mathcal{X}, \mathcal{B}_{\mathcal{X}})$ is a standard Borel space. A self-coupled MDP of M is a tuple $(\tilde{\mathcal{X}}, \mathcal{A}, \tilde{r}_c, \tilde{\mathcal{P}}, \gamma)$ where*

- *$\tilde{\mathcal{X}}$ is the product space $\mathcal{X} \times \mathcal{X}$, where $(\tilde{S}, \mathcal{B}_{\tilde{\mathcal{X}}})$ is the Borel product space $(\mathcal{X} \times \mathcal{X}, \mathcal{B}_{\mathcal{X}} \times \mathcal{B}_{\mathcal{X}})$*
- *$\tilde{r}_c{}^a : \mathcal{X} \times \mathcal{X} \to [0, 1]$ is the reward function such that $\tilde{r}_c{}^a([\mathbf{x}, \mathbf{y}]) = (1 - c)|r^a(\mathbf{x}) - r^a(\mathbf{y})|$*
- *$\tilde{\mathcal{P}} = (\tilde{\mathcal{P}}^a)_{a \in \mathcal{A}}$, such that $\tilde{\mathcal{P}}^a$ is a self-coupling $\lambda(\mathcal{P}^a, \mathcal{P}^a) \in \Lambda(\mathcal{P}^a, \mathcal{P}^a)$.*

Ferns and Precup (2014) provide the following theorem relating bisimulation metrics with the value of self-coupled MDPs

Theorem 3. ***Value Bounds for Self-Coupled MDPs (Ferns et al., 2004; Kemertas and Jepson, 2022)*** *Given an MDP $(\mathcal{X}, \mathcal{A}, r, \mathcal{P}, \gamma)$ and a bisimulation metric $d_{\sim,c}$, let $V^*(K)$ be the optimal value of the self-coupled MDP $(\tilde{\mathcal{X}}, \mathcal{A}, \tilde{r}_c, \tilde{\mathcal{P}}, \gamma)$ with $\tilde{\mathcal{P}} = K$. Then*

$$\forall a \in \mathcal{A} \text{ there exists a } \tilde{\mathcal{P}}^a, \text{ such that } d_{\sim,c} = V^*_c(\tilde{\mathcal{P}}^a) = \min_{K \in \Lambda(P,P)} V^*_c(K)$$

i.e., there exists a self-coupled MDP such that the bisimulation metric $d_\sim$ for the original MDP M corresponds to the optimal value function of the self-coupled MDP.

While the properties of bisimulation metrics described above provide connections to the optimal values of an MDP, we are often more interested in the values of a specific policy; particularly the learned policy at a given iteration of an RL algorithm. This necessitates an on-policy version of these metrics (Castro, 2020), which we now discuss.

2.4.4 On-Policy Bisimulation Metric

Having to enumerate over $\mathcal{X}$ makes $d^\sim$ intractable for continuous state spaces. Moreover, the maximization over actions makes it overly

pessimistic since bisimulation metrics measure the worst case distance between states. This is particularly undesirable if large worst case distances arise simply due to actions rarely executed under a policy (Castro, 2020), as illustrated in Figure 2.5. The *on-policy* bisimulation metric d^π resolves[14] these issues by swapping the maximization over actions with expectations over policy-induced transitions (Castro, 2020). It is the unique fixed point of the following iteration:

$$d^\pi(\mathbf{x}, \mathbf{y}) = |\mathbb{E}_\pi(r_\mathbf{x}) - \mathbb{E}_\pi(r_\mathbf{y})| + \gamma \mathcal{W}_1(\mathcal{P}^\pi_\mathbf{x}, \mathcal{P}^\pi_\mathbf{y} \mid d^\pi) \tag{2.3}$$

The on-policy bisimulation metric d^π also essentially retain the same properties as that for $d_\sim$ above – just with reference to values of a particular policy π (Castro, 2020). Most importantly,

1. The π-bisimulation operator converges, and the d^π is its unique fixed point.

2. Value functions are Lipschitz continuous with respect to d^π

$$|V^\pi(\mathbf{x}) - V^\pi(\mathbf{y})| \leq d^\pi(\mathbf{x}, \mathbf{y})$$

2.4.5 *Applications of Bisimulation Metrics*

Bisimulation metrics have primarily been used in the reinforcement learning community to obtain value-informed representations to accelerate policy learning. Since the bisimulation metric bounds differences in value functions, it induces value-based clusters in the representation space which can improve generalization across similar states. While early applications focused on state aggregation in finite MDPs (Ferns et al., 2004, 2006), these metrics are also applied on continuous MDPs (Ferns et al., 2011) in conjunction with modern value and policy learning algorithms (Kemertas and Jepson, 2022; **?**). The representations induced by bisimulation metrics have been shown to demonstrate invariance to policy-irrelevant distractors to improve generalization and safety (Zhang et al., 2020; Wang et al., 2025; Agarwal et al., 2021a; Liu et al., 2023). Castro and Precup (2010) propose bisimulation metric based representation learning as a means of policy transfer in MDPs. Rezaei-Shoshtari et al. (2024) connect bisimulation metrics to group fairness in RL to learn fair RL policies that encourage demographic parity fairness.

Recent work in bisimulation metrics has also focused on devising alternative formulations and empirical estimations of the metric to improve computational and sample efficiency. In Chapter 9, we provide an overview of these methods, and also propose a locality-preserving representation learning algorithm that can be applied

[14] Taylor et al. (2008) proposed mitigating this pessimism using the *lax* bisimulation equivalence (and its corresponding metric) that sets states to be equivalent if the rewards and transitions for two states match for potentially different actions. The on-policy metric by Castro (2020) not only reduces pessimism, but also provides a formalism to discuss metrics with respect to a given policy

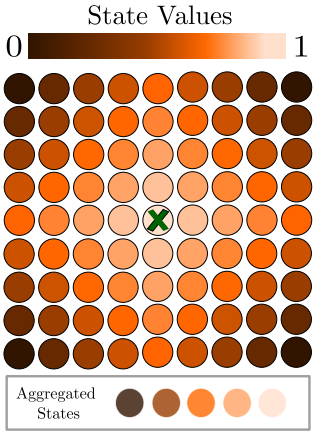


Figure 2.5: **The Standard Bisimulation Metric Leads to Pessimistic State Aggregation** Bisimulation metrics allow state-aggregation with respect to state-values. Here, the value function (top) is symmetric around the horizontal, vertical, and diagonal axes intersecting at the goal. Even though there are essentially 4 distinct states (bottom) in terms of their value, the standard bisimulation metric fails to capture this state aggregation. For instance, under the optimal policy, the bottom-left cell has exactly the same value as the top-right cell – just corresponding to different actions.

across these existing algorithms. Specifically, we propose learning representaions as behavioral eigenmaps (BeigeMaps), neural eigenfunctions of the Laplacian induced by bisimulation metrics.

Part II

Metric Informed Quantum Graphical Models

3
Introduction

3.1 Introduction

Over the years, there has been some interest in 'quantum-inspired' probabilistic models that incorporate mathematical formalisms from quantum mechanics into classical machine learning. Some of the early quantum-inpsired models proposed in the literature included quantum graphical models (QGMs) that utilize quantum mechanical state representations and dynamics for classical Bayesian inference (Leifer and Poulin, 2008; Yeang, 2010). Hidden quantum Markov models (HQMMs) are one of the more well-investigated of these QGMs, and are known to be more expressive than classical hidden Markov models (Monras et al., 2010; Srinivasan et al., 2018d).

Another popular class of probabilistic models includes quantum tensor networks known as matrix product states (MPS) (Perez-Garcia et al., 2006), which are equivalent to tensor-train decompositions (Oseledets, 2011). Originally used to represent systems of quantum particles, MPS have recently been applied to classical sequential modeling (Stoudenmire and Schwab, 2016; Han et al., 2018). Similar proposals have also been studied in *classical* stochastic processes and weighted automata literature, with little work on formally connecting them. Namely, observable operator models (Jaeger, 2000) or predictive state representations (PSRs) (Singh et al., 2004) from the machine learning community, and stochastic weighted automata (SWA) (Balle et al., 2014b) from the formal languages community have remarkably similar structure to HQMMs and MPS. While the connections between PSRs and SWA are well-established, similar formal connections are yet to be made with MPS and HQMMs.

We use HQMMs as our primary model of interest and focus on learning these quantum-inspired models from data, and examining its expressiveness in relation to other seemingly disparate models from stochastic processes, weighted automata, and quantum tensor networks. Towards both ends, we will rely on utilizing and analyzing

the metric structure of HQMM states and parameters, illustrating how this structure can help devise efficient learning algorithms and better understand this model class.

Outline In this chapter, we provide an introduction to HQMMs, the quantum inspired generalizations of Hidden Markov Models (HMMs). In Section 3.5 and Section 3.6, we compare the geometry of the state spaces (polyhedral cones and spectraplexes) and operator spaces (multinomial manifolds and Stiefel manifolds) of these model classes to get a deeper understanding of their differences. Finally, we turn to the critical question: does the additional complexity introduced by HQMMs actually yield any meaningful improvements over HMMs? In Section 3.7, we summarize the positive answer to this question found in the literature in the context of model expressiveness: there exist processes that can be modeled exactly by a finite dimensional HQMM that cannot be modeled by a finite dimensional HMM.

Notation We use bold-face capitalized letters for matrix and tensor operators (e.g. $\mathbf{A}$), bold-face lower case letters for vectors (e.g. $\mathbf{x}$) and plain non-bold symbols for scalars. Vector-arrows are used to indicate vectorization (column-first convention) of matrices (e.g. $\vec{\mathbf{A}}$). We use $\mathbb{R}$ and $\mathbb{C}$ to denote the real and complex scalar fields. 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 ($\bar{\mathbf{A}}^T = \mathbf{A}^\dagger$). Note that we also use overhead bars to denote a sequence of observations (e.g. $\bar{y}$) without conjugation. Finally, $\mathrm{Tr}(\cdot)$ denotes the trace operation applied to matrices, and $\otimes$ denotes the Kronecker product, and the bra-ket notation $\langle \cdot, \cdot \rangle$ denotes an inner-product.

3.2 *Hidden Markov Models (HMMs)*

Hidden Markov Models (HMMs) are a well-known class of statistical models of sequential stochastic processes. Developed in the 1960s (Baum and Petrie, 1966; Baum and Eagon, 1967), HMMs provide an intuitive means of modeling sequential processes under the Markov assumption: the future is independent of the past given the current state. Despite – or perhaps because of – its relative simplicity, HMMs have found wide-ranging applications ranging from speech-recognition (Rabiner, 1989) to bioinformatics (Li and Stephens, 2003; Ernst and Kellis, 2012) to optimal control (Doucet et al., 2009). In these applications, HMMs are used to model sequential processes

where the internal state of the system is not directly observable, but must be inferred from sequences of potentially noisy observations. Given reasonable models of state transitions and observation emissions, HMMs can be learned efficiently from data using various algorithms including the popular Baum-Welch Expectation Maximization (Baum et al., 1972) algorithm.

Formally, we define a Hidden Markov Model as follows

Definition 10 (HMMs)**.** *An n-dimensional Hidden Markov Model with a set of discrete observations $\mathcal{O}$ is a tuple ($\mathbb{R}^n$, $\mathbf{A}$, $\mathbf{C}$, $\mathbf{x}_0$) where initial state $\mathbf{x}_0$, transition matrix $\mathbf{A}$, and emission matrix $\mathbf{C}$ satisfy the following conditions:*

(i) Non-negative parameters: $\mathbf{x}_0 \in \mathbb{R}^n_{\geq 0}$, $\mathbf{A} \in \mathbb{R}^{n\times n}_{\geq 0}$, $\mathbf{C} \in \mathbb{R}^{|\mathcal{O}|\times n}_{\geq 0}$,

(ii) Normalized initial state: $\vec{\mathbb{1}}^T\mathbf{x}_0 = 1$,

(iii) Column-stochastic operators: $\vec{\mathbb{1}}^T\mathbf{A} = \vec{\mathbb{1}}^T\mathbf{C} = \vec{\mathbb{1}}^T$.

$\vec{\mathbb{1}} = [1, \cdots, 1]$ is a vector with 1 in all entries.

HMM states $\mathbf{x}$ are also known as *belief* states, and are interpretable as probability distributions over hidden system states; the i-th element of the state vector $\mathbf{x}_0$ corresponds to the probability or the "belief" that the system is in the i-th latent state. This explains the need for non-negativity and normalization constraints on HMM states.

At each time-step, we update the belief state and condition on observation using the column-stochastic matrices $\mathbf{A}$ and $\mathbf{C}$ respectively:

$$\mathbf{x}'_t = \mathbf{A}\mathbf{x}_{t-1} \qquad \mathbf{x}_t = \frac{\text{diag}(\mathbf{C}_{(y,:)})\mathbf{x}'_t}{\vec{\mathbb{1}}^T\text{diag}(\mathbf{C}_{(y,:)})\mathbf{x}'_t}, \tag{3.1}$$

where $\text{diag}(\mathbf{C}_{y,:})$ places the y-th row of matrix $\mathbf{C}$ in a diagonal matrix. Note that the y-th row of the $\mathbf{C}$ corresponds to the y-th observation $y \in \mathcal{O}$.

We can compute the probability of a sequence of observations $\bar{y} = y_1, \ldots, y_t$ for $y_i \in \mathcal{O}$ from a given belief state $\mathbf{x}$ as follows:

$$P(\bar{y}) = \vec{\mathbb{1}}^T\text{diag}(\mathbf{C}_{(y_t,:)})\mathbf{A} \cdots \text{diag}(\mathbf{C}_{(y_1,:)})\mathbf{A}\mathbf{x} \tag{3.2}$$

The state updates and probability computations above can be simplified to some extent by defining a new set of operators[1] $\{\mathbf{T}_y = \texttt{diag}(\mathbf{C}_{(y,:)}\mathbf{A}\}$ that combine both $\mathbf{A}$ and $\mathbf{C}$ operators as follows

$$\mathbf{x}_t = \frac{\mathbf{T}_y\ \mathbf{x}_{t-1}}{\vec{\mathbb{1}}^T\mathbf{T}_y\ \mathbf{x}_{t-1}} \qquad P(\bar{y}) = \vec{\mathbb{1}}^T\mathbf{T}_{y_t} \cdots \mathbf{T}_{y_1}\mathbf{x} \tag{3.3}$$

[1] This corresponds to the Observable Operator Model (OOM) formulation of HMMs, and $\{\mathbf{T}_y\}$ are called observable operators. We discuss OOMs in detail in Chapter 5.

Zooming out of its specifics, an HMM basically consists of a representation of probability distributions (states) and a set of linear operators that evolve these states over time conditioned on observations. The interest in using 'quantum-inspired' graphical models

as alternatives to HMM stems from the fact that quantum particles are modeled as probability distributions over discrete latent states, which evolve over time in response to measurements (observations) via linear operators. A natural curiosity thus arises: what if we encode *classical* data into the states of a quantum particle, and assign quantum-measurement operators as the means of Markovian evolution over time? This is the idea behind Hidden Quantum Markov (HQMM) models (Monras et al., 2010), which we now discuss in the next section.

We begin with a brief introduction to quantum states and quantum channels that underlie these models[2].

[2] For a thorough introduction to these and other concepts in quantum information, we refer the reader to Nielsen and Chuang (2011). Wood et al. (2015a) and Miszczak (2011) provide instructive analyses of quantum channels and their various representations.

3.3 *Quantum Channels*

Quantum Wavefunctions The state of a quantum particle is commonly encoded as a compledx valued wavefunction[3] ψ, which is often written as a *ket* operator[4] $|\psi\rangle$. As with HMMs, the state of a quantum system encodes the probability distribution over a discrete latent space. For instance, a qubit, the atomic unit of information in quantum information, can be measured to be in one of two states: 0 or 1. The belief over these measurements is encoded as a 2-dimensional wavefunction $|\psi\rangle$, where the i-th element encodes the probability of the qubit being measured to be in the i-th latent state. However unlike HMMs, this encoding is not simply the identity operation.

[3] Not all quantum states can be represented by a single wavefunction; the states that can are known as *pure* quantum states. We discuss *mixed* quantum states in the next section

[4] The complex-conjugate of the ket operator is the *bra* operator $\langle\psi|$. Inner products are then denoted as *brakets* $\langle\psi, \psi\rangle$. Outer-products are written similarly as $|\psi\rangle\langle\psi|$.

Given an n-dimensional wavefunction $|\psi\rangle \in \mathbb{C}^n$, the probability that the particle is in the i-th latent state is computed as $||\psi\rangle[i]|^2$. Consequently, the L_2 norm of a valid quantum state must be equal to one. Specifically,

$$P(y_i) = ||\psi\rangle[i]|^2 \qquad \||\psi\rangle\|^2 = \langle\psi, \psi\rangle = 1$$

Another consequence of this mapping to probabilities is that the elements of $|\psi\rangle$, known as *probability amplitudes*, are not restricted to be positive – in fact, they are complex valued. This seemingly minor relaxation ends up having profound implications in quantum mechanics. The joint state of two quantum states can be formed through constructive and destructive interference. Constructive interference is simply the addition of individual probability amplitudes, and is the only type of interaction possible if all probability amplitudes were non-negative real numbers. But without the non-negativity restriction, quantum states can also interfere destructively, e.g., the negative probability amplitudes from one particle can cancel out the probability amplitude for the corresponding latent-state from a different quantum particle. While the physical implications of destructive in-

terference are outside the scope of this thesis, we do note that many 'non-intuitive' phenomena in quantum-mechanics ultimately boil down to interference, and thus non-negative probability amplitudes.

Density Operators Not all quantum states can be neatly represented by a wavefunction $|\psi\rangle$. Quantum state that can be encoded by a single wavefunction are known as *pure* states. Quantum systems can also exist as probabilistic mixtures of pure states or *mixed* states. To encompass both pure and mixed states, we turn to a more general formulation of quantum states called *density operators*. An n-dimensional density operator $\hat{\boldsymbol{\rho}} \in \mathbb{C}^{n \times n}$ is a complex-valued valued matrix that is positive semi-definite (PSD) and has unit-trace.

Just as the entries of a wavefunction encode the probability amplitudes of beliefs over latent states, the i-th diagonal entry of $\hat{\boldsymbol{\rho}}$ corresponds to the probability of the i-th latent state. Note that unlike the wavefunction, the diagonals list the actual probabilities, not the probability amplitudes. Since $\hat{\boldsymbol{\rho}}$ is PSD, we know that its diagonals are always non-negative real-values, and the unit-trace requirement ensures that these probabilities are normalized.

Quantum Channels Let $\hat{\boldsymbol{\rho}}$ denote the state of a quantum system. If $\hat{\boldsymbol{\rho}}$ is a *closed*[5] system, the time-evolution of this state can always be defined using a unitary operator $\{\mathbf{U} \in \mathbb{C}^{n \times n} : \mathbf{U}^\dagger \mathbf{U} = \mathbb{I}\}$ as follows:

$$\hat{\boldsymbol{\rho}}' = \mathcal{E}(\hat{\boldsymbol{\rho}}) = \mathbf{U}\hat{\boldsymbol{\rho}}\mathbf{U}^\dagger$$

where $\mathcal{E}$ denotes a quantum operation.

However, if the quantum system is *open*, unitary transformations[6] are no longer sufficient to describe the evolution of $\hat{\boldsymbol{\rho}}$. To model $\hat{\boldsymbol{\rho}}$ as part of an open quantum system, we can introduce an additional *environment* particle $\hat{\boldsymbol{\rho}}_{\text{env}}$. This environment particle covers everything else in the system that is not encoded in $\hat{\boldsymbol{\rho}}$, hence the composite system $\hat{\boldsymbol{\rho}} \otimes \hat{\boldsymbol{\rho}}_{\text{env}}$ forms a closed system.

While the composite closed system $\hat{\boldsymbol{\rho}} \otimes \hat{\boldsymbol{\rho}}_{\text{env}}$ always undergoes a unitary transformation, the constituent sub-systems, including $\hat{\boldsymbol{\rho}}$ can undergo non-unitary transformations. The state of an individual sub-system in such a composite system is obtained by taking a partial trace with respect to the other particle. Finally, as with any closed system, the evolution of this composite system is modeled as a unitary transformation $\mathbf{U}(\hat{\boldsymbol{\rho}} \otimes \hat{\boldsymbol{\rho}}_{\text{env}})\mathbf{U}^\dagger$. Thus, the evolution of an open quantum system is modeled as

$$\hat{\boldsymbol{\rho}}' = \mathcal{E}(\hat{\boldsymbol{\rho}}) = \text{Tr}_{\text{env}}\left[\mathbf{U}(\hat{\boldsymbol{\rho}} \otimes \hat{\boldsymbol{\rho}}_{\text{env}})\mathbf{U}^\dagger\right], \tag{3.4}$$

where Tr_{env} is the partial trace with respect to $\hat{\boldsymbol{\rho}}_{\text{env}}$.

[5] An open quantum system is one that is influenced by the external environment around it. If the external environment is also incorporated into the system, we obtain a closed system

[6] Intuitively, a unitary transformation preserves total probability. But in an open system, some probability "leaks out" from $\hat{\boldsymbol{\rho}}$ to the rest of the environment.

In quantum information theory, both open and closed systems can be described using *quantum channels*, linear mappings from one valid quantum state to another (Nielsen and Chuang, 2011). The above equation is one representation of such a quantum-channel, commonly known as the *Stinespring representation* (Wood et al., 2015b) as it can be viewed as an application of the Stinespring dilation theorem (Stinespring, 1955).

Operator-Sum Representation of Quantum Channels The Stinespring representation (Equation 3.4) of a quantum channel with respect to a fictitious environment particle provides a physically intuitive understanding of quantum dynamics. But an alternate formulation, known as the Kraus-representation or the *operator-sum* representation tends to be easier to work with, especially when designing a learning algorithm. Specifically, a quantum channel can be parameterized by a set of *Kraus operators* $\{\mathbf{K}_w\}$ (Kraus, 1971). The state-update in Equation 3.4 can be equivalently expressed using these operators as follows:

$$\hat{\boldsymbol{\rho}}' = \mathcal{E}(\hat{\boldsymbol{\rho}}) = \sum_w \mathbf{K}_w \hat{\boldsymbol{\rho}} \mathbf{K}_w^\dagger \qquad \text{s.t.} \quad \sum_w \mathbf{K}_w^\dagger \mathbf{K}_w = \mathbb{I} \tag{3.5}$$

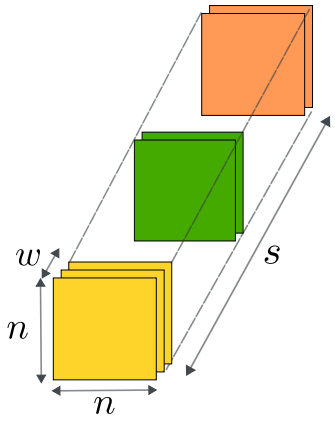


Figure 3.1: **POVM Operators for an HQMM**: Given a finite observation space $\mathcal{O}$ with s observations, we can define a set of Kraus operators $\{\mathbf{K}_{w_y}\}$ for each observation y, here denoted with matrices of different colors. Each Kraus operator of the set (indexed by w_y), is an $n \times n$ matrix.

Measurement of Quantum Systems In quantum mechanics, sampling from a belief state corresponds to making a measurement or an observation of the state. A simple paradigm for quantum measurement is known as Projection Valued Measurement (PVM), under which, the probability of observing $\hat{\boldsymbol{\rho}}$ to be in the y-th state is given by $\text{Tr}(\mathbf{P}_y \hat{\boldsymbol{\rho}})$, where $\mathbf{P}_y = \mathbf{P}_y^2$ is a projection operator that projects $\hat{\boldsymbol{\rho}}$ onto the eigenbasis of the s-th state. A significant drawback of PVMs is that they can only describe measurements in a closed system, but not open systems where measurements are performed on a larger system (e.g. the experiment environment) of which $\hat{\boldsymbol{\rho}}$ is a sub-system. Such measurements can be modeled using a different paradigm known as Positive Operator Valued Measurements (POVMs), which utilize the operator-sum-representation of quantum channels.

A POVM is defined by a set of PSD matrices $\{\mathbf{F}_y\}$ constrained by the normalization condition $\sum_i \mathbf{F}_y = \mathbb{I}$. The probability of observing y when the system is in state $\hat{\boldsymbol{\rho}}$ is computed as

$$\begin{gathered} \Pr(y \mid \hat{\boldsymbol{\rho}}) = \text{Tr}(\mathbf{F}_y \hat{\boldsymbol{\rho}}) \\ \text{POVM operators}: \ \{\mathbf{F}_y \mid \mathbf{F}_y \text{ is P.S.D}\} \ \text{ such that } \ \sum_y \mathbf{F}_y = \mathbb{I} \end{gathered} \tag{3.6}$$

It is easy to see that PVMs are a special case of POVMs where $\mathbf{F}_y$ are constrained to be projection matrices.

We can further expand POVMs to describe the evolution of a quantum state following a measurement. As every POVM operator is PSD, we can decompose it as $\mathbf{F}_y = \sum_{w_y} \mathbf{K}_{w_y}^\dagger \mathbf{K}_{w_y}$, where w_y is a variable number of operators for every observation y, as visualized in Figure 3.1. The Kraus matrices $\{\{\mathbf{K}_{w_y}\}\}_y$ describe the state transition from $\hat{\boldsymbol{\rho}}$ to $\hat{\boldsymbol{\rho}}'$ following an observation y as follows

$$\begin{aligned} \Pr(y|\hat{\boldsymbol{\rho}}) &= \operatorname{Tr}\left(\mathbf{K}_y^\dagger \mathbf{K}_y \hat{\boldsymbol{\rho}}\right) = \operatorname{Tr}\left(\mathbf{K}_y \hat{\boldsymbol{\rho}} \mathbf{K}_y^\dagger\right) \\ \hat{\boldsymbol{\rho}}' &= \frac{\mathbf{K}_y \hat{\boldsymbol{\rho}} \mathbf{K}_y^\dagger}{\Pr(y|\hat{\boldsymbol{\rho}})} = \frac{\mathbf{K}_y \hat{\boldsymbol{\rho}} \mathbf{K}_y^\dagger}{\operatorname{Tr}\left(\mathbf{K}_y \hat{\boldsymbol{\rho}} \mathbf{K}_y^\dagger\right)} \end{aligned} \tag{3.7}$$

where $\mathbf{K}^\dagger$ is the conjugate-transpose of $\mathbf{K}$. Hidden quantum Markov models (Monras et al., 2010) were originally formulated using the above Kraus representation of POVM operators for measurements, which we now describe.

3.4 *Hidden Quantum Markov Models (HQMMs)*

We present the original formulation of HQMMs as Kraus-HQMMs (K-HQMMs) as follows:

Definition 11 (K-HQMMs)**.** *An n-dimensional Kraus-Hidden Quantum Markov Model with a set of discrete observations $\mathcal{O}$ is a tuple $(\mathbb{C}^{n\times n}, \{\mathbf{K}_{y,w_y}\}_{y\in\mathcal{O}}, \hat{\boldsymbol{\rho}}_0, Tr(\cdot))$ where initial state $\hat{\boldsymbol{\rho}}_0 \in \mathbb{C}^{n\times n}$ and Kraus operators $\{\mathbf{K}_{y,w_y}\}_{y\in\mathcal{O}, w_y\in\mathbb{N}} \in \mathbb{C}^{n\times n}$ satisfy the following constraints:*

(i) $\hat{\boldsymbol{\rho}}_0$ *is a Hermitian PSD matrix of arbitrary rank,*

(ii) Normalized Initial State: $Tr(\hat{\boldsymbol{\rho}}_0) = 1$*,*

(iii) Normalized marginal over observations: $\sum_{y,w} \mathbf{K}_{y,w}^\dagger \mathbf{K}_{y,w} = \mathbb{I}$*.*

The state update after observing y is computed as

$$\hat{\boldsymbol{\rho}}_t = \frac{\sum_{w_y} \mathbf{K}_{y,w_y}\, \hat{\boldsymbol{\rho}}_{t-1}\, \mathbf{K}_{y,w_y}^\dagger}{\operatorname{Tr}\left(\sum_{w_y} \mathbf{K}_{y,w_y}\, \hat{\boldsymbol{\rho}}_{t-1}\, \mathbf{K}_{y,w_y}^\dagger\right)}, \tag{3.8}$$

and probability of a given sequence is given by:

$$P(\bar{y}) = \operatorname{Tr}\left(\sum_{w_{y_t}} \mathbf{K}_{y_t,w_{y_t}} \dots \left(\sum_{w_{y_t}} \mathbf{K}_{y_1,w_{y_1}} \hat{\boldsymbol{\rho}}_0 \mathbf{K}_{y_1,w_{y_1}}^\dagger\right) \dots \mathbf{K}_{y_t,w_{y_t}}^\dagger\right) \tag{3.9}$$

Both HMM and HQMM states satisfy notions of positivity and normalization. The constraints on states arise since they need to correspond to probability distributions. The constraints on operators are needed to ensure that evolution via operators generate new states that are still valid.

While quantum-states encode the state of quantum particles, they could just as easily be used to encode the states of classical probabilistic sequential dynamics with discrete state spaces. This is essentially the central motivation behind HQMMs, particularly when applied to classical machine learning. Viewed this way, HQMMs appear as an alternative to traditional HMMs, wherein quantum density operators replace traditional state vectors, and quantum channels replace HMM transition and emission operators. This raises the natural question: what do we gain and lose when swapping classical HMMs with quantum-inspired HQMMs?

For HQMMs to be considered as useful alternatives to HMMs, we could reasonably require that HQMMs should allow us to model some processes that cannot be modeled by HMMs. If not, they should at least allow us to model the same processes as HMMs but more efficiently. Indeed, if HQMMs do not provide any additional expressiveness or efficiency, it is difficult to justify the additional mathematical complexity introduced by quantum dynamical systems. As we now discuss, this property is, in fact, satisfied: HQMMs form a superset of HMMs – any process that can be modeled using a finite-dimensional HMM, can also be modeled using a finite-dimensional HQMMs (Monras et al., 2010) but not vice-versa. In fact, empirical results suggest that HQMMs tend to require fewer latent dimensions[7] to model the same process (Srinivasan et al., 2018e; Adhikary et al., 2020).

[7] not to be confused with number of parameters. A lower-dimensional HQMM can have more parameters than a higher dimensional-HMM

To analyze the relative expressiveness of HMMs and HQMMs, let us begin by taking a closer look at the geometric structure of their state and operator spaces.

3.5 *State Space Geometry*

HMM State Space Geometry The critical constraint defining the structure of HMM state spaces is that any HMM state must directly correspond to a probability distribution. This translates to a non-negativity constraint (since probabilities cannot be negative) and a normalization constraint (since probabilities must sum to 1).

For HMMs, the non-negativity constraint on states is enforced in arguably the strictest way possible – every element of the state vector is required to be non-negative. The state-normalization constraint can then be implemented as requiring HMM states to have unit 1-norm, i.e., its elements must sum to 1. The state space carved out by these constraints can be viewed as a pointed polyhedral cone (Jaeger, 2000). As shown in Figure 3.2, a polyhedral cone can be viewed as constructing an approximation of smooth (non-polyhedral) cone by stitching together multiple triangles that all meet at a common tip.

The corners where these triangular sections meet form sharp edges that define the borders of the cone. The number of such sharp edges is determined by the number latent states. Indeed, with an infinite number of latent states, we would obtain an exact smooth cone – but a finite number of states can only provide an approximation.

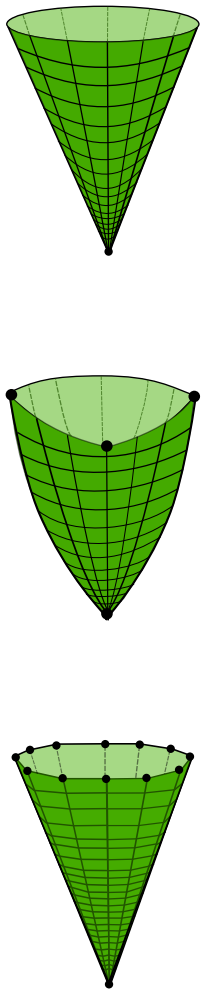


Figure 3.2: **Illustrations of State Space Geometrices of HMMs and HQMMs** Non-polyhedral cones (top), Spectraplexes (middle), and Polyhedral cones (bottom).

HQMM State Space Geometry As with HMMs, HQMM states also satisfy non-negativity and normalization constraints to ensure that they correspond to valid probability distributions. The non-negativity constraint is enforced by requiring the state matrix $\hat{\boldsymbol{\rho}}$ to be positive semi-definite. Note that individual elements of the state-matrix are not required to be non-negative – in fact, they are not even required to be real-valued. Recall that PSD matrices have real and non-negative diagonals, and the diagonals of $\hat{\boldsymbol{\rho}}$ list the probabilities of individual latent states. Finally, the normalization constraint is enforced by requiring $\hat{\boldsymbol{\rho}}$ to have unit-trace, i.e., the diagonal entries must sum to 1.

Similar to HMMs, the state space of HQMMs consisting of unit-trace PSD matrices also carve out a pointed convex cone, but with different goemetric properties. Specifically, 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*.

The polyhedral cones of HMM states and spectraplexes of HQMMs define the set of all *possible* valid states. Technically, these states form all possible valid *initial* states, since some states may not be reachable unless the process starts off that that state. Given an initial state, the portion of the state space that is actually reachable is defined by the operators of the model. Since the state-spaces of both HMMs and HQMMs are constrained, their operators must also be constrained in a way to avoid generating states outside the valid space. We now discuss these constraints, and how we can visualize them.

3.6 *Operator Space Geometry*

As with states, HMM and HQMM operators also satisfy non-negativity and normalization constraints.

HMM Operator Space Geometry For HMMs, the non-negativity constraint on operators is enforced by requiring all their elements to be non-negative – the same constraint placed on states. The normalization constraint is satisfied by requiring both transition and emission operators to be column-stochastic. Note that, again, this is the same constraint as for HMM states – just imposed over all individual

columns of operators.

Geometrically, the valid space of HMM operators can be characterized as a Multinomial manifold:

$$\text{Multinomial Manifold: } \{\mathbf{X} \in \mathbb{R}^{m \times n} : \mathbf{X}_{ij} > 0 \ \forall i, j \ \ \mathbf{X}^T \mathbf{1}_m = \mathbf{1}_n\}$$

HQMM Operator Space Geometry Compared to the strict non-negativity constraints of HMM operators, the definition of HQMMs in Definition 11 might, at first glace, suggest that there are no non-negativity constraints enforced on HQMM operators. In contrast, HQMM operators actually satisfy a relatively strong positivity constraint – called *complete positivity* – but we end up getting this property 'for free' with the Kraus operators from the operator-sum representation of quantum channels.

Specifically, the non-negativity constraint on HQMM operators are basically designed to ensure that the non-negativity of HQMM states $\hat{\boldsymbol{\rho}}$ is invariant to these operations. In other words, we need HQMM operators to preserve the positive semi-definiteness of $\hat{\boldsymbol{\rho}}$. But the quantum-mechanical interpretation of quantum channels requires more – it must also account for any other auxiliary states. Specifically, it is not sufficient that the quantum channel $\mathcal{E}$ preserves the positive semi-definiteness of $\hat{\boldsymbol{\rho}}$ – it must also preserve the positive semi-definiteness of the joint state of $\hat{\boldsymbol{\rho}}$ in conjunction with its environment. Namely, any quantum particle $\hat{\boldsymbol{\rho}}$, can always be viewed as part of a larger composite-system $\hat{\boldsymbol{\rho}} \otimes \hat{\boldsymbol{\rho}}_e$, where $\hat{\boldsymbol{\rho}}_e$ includes all other particles in the environment. Since the HQMM operation $\mathcal{E}$ should only update $\hat{\boldsymbol{\rho}}$, it can be viewed as the composite operation $\mathcal{E} \otimes \mathbb{I}$ where $\hat{\boldsymbol{\rho}}_e$ undergoes the identity operation. Now, we definitely need the quantum channel acting on $\hat{\boldsymbol{\rho}}$ to preserve its positive semi-definiteness, i.e. $\hat{\boldsymbol{\rho}} \succeq 0 \implies \mathcal{E}(\hat{\boldsymbol{\rho}}) \succeq 0$. But we also require that the final composite system also be PSD, i.e., $\hat{\boldsymbol{\rho}} \succeq 0, \hat{\boldsymbol{\rho}}_e \succeq 0 \implies (\mathcal{E} \otimes \mathbb{I})(\hat{\boldsymbol{\rho}} \otimes \hat{\boldsymbol{\rho}}_e) \succeq 0$, for any $\hat{\boldsymbol{\rho}}_e$ of arbitrary dimensions.

The set of operators that satisfy the strict positivity required described above are known as Completely Positive (CP) operators. While it is highly non-trivial to define constructive constraints to ensure complete-positivity, we actually get it for 'free' via the Kraus operator representation of K-HQMMs since any operation of the form $\sum_w \mathbf{K}_w \hat{\boldsymbol{\rho}} \mathbf{K}_w^\dagger$ is guaranteed to be CP. In the next chapter, we will reformulate HQMMs using a different formulation, and discuss alternative means of enforcing this constraint.

In addition to complete positivity, HQMM operators must also ensure that they always return Hermitian states, i.e., they must preserve the Hermitian property $\hat{\boldsymbol{\rho}} = \hat{\boldsymbol{\rho}}^\dagger$ of HQMM states. As with the CP

constraint, the Kraus operator formulation satisfies this constraint by design: for any PSD matrix $\mathbf{M}$, we have $\left(\sum_i \mathbf{A}_i^\dagger \mathbf{M} \mathbf{A}_i\right)^\dagger = \sum_i \mathbf{A}_i^\dagger \mathbf{M} \mathbf{A}_\mathbf{i}$.

The normalization constraint on HQMM-operators or the *trace-preservation* (TP) constraint is the only one of the CP-HP-TP constraint-triple that is not trivially satisfied by K-HQMM operators. The TP constraint requires that the operators be trace-invariant, i.e., the trace of the input state $\hat{\boldsymbol{\rho}}$ remains unchanged after the operation. This constraint can be enforced by requiring that $\sum_{y,w} \mathbf{K}_{y,w}^\dagger \mathbf{K}_{y,w} = \mathbb{I}$ (Nielsen and Chuang, 2011).

The TP constraint can be simplified to some extent by introducing a new operator $\boldsymbol{\kappa}$ formed by vertically stacking the individual Kraus operators i.e. $\boldsymbol{\kappa}^T = [\mathbf{K}_{w_{1y_1}} | \cdots | \mathbf{K}_{w_{my_1}} | \cdots | \mathbf{K}_{w_{1y_n}} | \cdots | \mathbf{K}_{w_{my_n}}]$, as visualized in Figure 3.3. The TP constraint is then translated to an orthonormality constraint on $\boldsymbol{\kappa}$ as follows

$$\boldsymbol{\kappa}^\dagger \boldsymbol{\kappa} = \mathbb{I} \iff \sum_{w,y} \mathbf{K}_{w,y}^\dagger \mathbf{K}_{w,y} = \mathbb{I}$$

This constraint defines a space of operators $\boldsymbol{\kappa}$ with a specific metric structure – namely $\boldsymbol{\kappa}$ is an element of a Riemannian manifold known as the *Stiefel* manifold. This manifold is essentially the same as the space of left-unitary matrices, but more general as it includes non-square matrices, defined more formally as:

$$\text{Stiefel Manifold:} \{\mathbf{X} \in \mathbb{C}^{m \times n} : \mathbf{X}^\dagger \mathbf{X} = \mathbb{I}\}$$

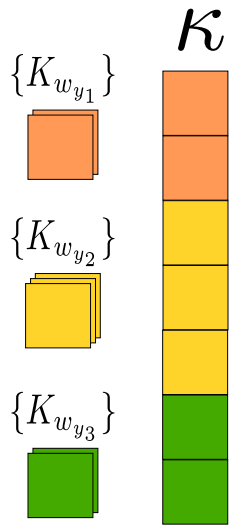


Figure 3.3: **Stacked Kraus operators on the Stiefel manifold** The $\boldsymbol{\kappa}$ matrix formed by vertically stacking all Kraus operators satisfies $\boldsymbol{\kappa}^\dagger \boldsymbol{\kappa} = \mathbb{I}$

3.7 *Relative Expressiveness of HMMs and HQMMs*

Having characterized the geometries of both the state and operator spaces of HMMs and HQMMs, we now turn to relative expressiveness of these models. Specifically, we use the following definition of relative expressiveness[8]:

Definition 12 (Relative Expressiveness of Models)**.**

- $\mathbf{A} \subseteq \mathbf{B}$*: A model class B is at least as expressive as model class A, i.e., $A \subseteq B$, if all finite-dimensional models in class A can be modeled exactly by a finite-dimensional model in class B.*
- $\mathbf{A} \subset \mathbf{B}$*: Moreover, model class B is more expressive than model class A, i.e., $A \subset B$, if $A \subseteq B$ and there exists a finite-dimensional model in class B that cannot be modeled exactly by a finite-dimensional model in class A.*

Relative Expressiveness: HMM $\subset$ HQMM Using the above definition, we can characterize the relative expressiveness of HMMs and HQMMs by seeking a family of processes that separate the two

[8] Note that this is a binary comparison between model classes. For example, a model class would be more expressive than another even if there exists just one (potentially esoteric model) that cannot be modeled exactly by the other class.

classes. One such family is that of processes that require oscillations in the latent space dynamics (Zhao and Jaeger, 2010a).

The state-space trajectories of HMMs are generated through repeated applications of linear operators $\mathbf{T}_y$, as shown in Equation 3.3. Such sequential linear operations can generate oscillatory latent state trajectories only if the operators have complex-valued eigenvalues. However, since $\mathbf{T}_y$ are non-negative, they can only have real eigenvalues. Thus, HMM operators cannot induce oscillations in the latent state.

As with HMMs, the state updates in HQMMs are also linear with respect to the state-space, but unlike HMMs they can generate oscillatory behavior. This can be seen clearly if we simply vectorize[9] the update equations in Definition 11:

$$\mathsf{vec}(\hat{\boldsymbol{\rho}}) \propto \mathsf{vec}\big(\sum_w \mathbf{K}_{y_w} \hat{\boldsymbol{\rho}} \mathbf{K}_{y_w}^\dagger\big) \propto \big(\sum_w \mathbf{K}_{y_w}^\dagger \otimes \mathbf{K}_{y_w}\big)\mathsf{vec}(\hat{\boldsymbol{\rho}})$$

Here, the operator $\big(\sum_w \mathbf{K}_{y_w}^\dagger \otimes \mathbf{K}_{y_w}\big)$ is now simply a $n^2 \times n^2$ matrix. Unlike HMM operators, this operator is not restricted to be non-negative, and can have complex-valued eigenvalues. Thus, it is able to generate oscillations in the latent state. Such HQMMs cannot be modeled using a finite-dimensional HMM.

We can illustrate the separation between HMMs and HQMMs using a concrete example originally devised by Jaeger (2000) – the probability clock. This is a simple process defined over two observations $\mathcal{O} = \{a, b\}$ that induces oscillatory behavior in the conditional probability $P(b|a)$ as shown in Figure 3.5. Jaeger (2000) show that this requires similar oscillatory behavior in the latent states, and that finite-dimensional HMMs cannot model this process. It has been shown (Srinivasan et al., 2018e; Adhikary et al., 2020) that finite-dimensional HQMMs can, in fact, exactly model processes including the probability clock; demonstrating that HQMMs are more expressive than HMMs: HMM $\subset$ HQMM.

From a geometric perspective, Jaeger (2000) identify any stochastic process, satisfying a specific notion of finiteness[10], can be modeled by general class of models known as Observable Operator Models (OOMs). We discuss this general model class in Chapter 5 – indeed, we will demonstrate that OOMs contain HQMMs as well. For now, we note that the state-space of general OOMs define non-polyhedral convex cones, and HMMs are known to be a specific class of OOMs with polyhedral convex cones. The probability clock is a specific example of a process that cannot be modeled exactly on a finite-dimensional polyhedral cone.

Relative Compactness While our definition of relative expressiveness in Definition 12 captures whether one class is more expressive-

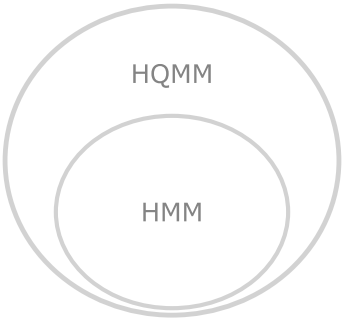


Figure 3.4: **HQMMs are more Expressive than HMMS** HMM $\subset$ HQMM

[9] The operators formed through this vectorization are known as Liouville operators. We use this vectorized formulation to define Liouville-HQMMs in Chapter 5

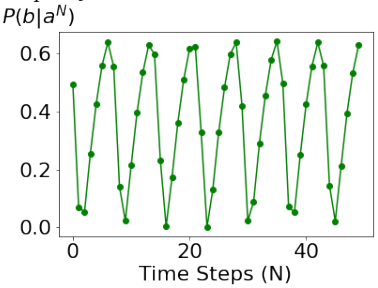


Figure 3.5: **The Probability Clock** The 'probability clock' process generated by a finite dimensional OOM. Finite dimensional HMMs are unable to model this oscillating behavior exactly.

[10] Specifically, Jaeger (2000) showed that OOMs can model any stochastic process with a 'systems-dynamics' matrix with finite rank. Note that this does not restrict the latent dimension to be finite.

ness than other, it does not include a notion of compactness. Depending on the application, we may be interested in different notions of compactness either in the parameter space or the state-space.

From a modeling perspective, it may be desirable to model a process using as few latent states as possible. Srinivasan et al. (2018a) have shown that any n-dimensional HMM can be modeled by an HQMM with no more than n^2 dimensions. However, it is not clear if this is a tight upper-bound. In fact, empirical evaluations have demonstrated multiple examples where an HQMM is able to model a process with fewer latent dimensions than an HMM. In Figure 3.6, we present results from our own experiments on synthetic data corresponding to the Stern-Gerlach experiment, an illustration of a widely studied quantum mechanical process. As illustrated in the results, 2-dimensional HQMMs were able to model the data with higher accuracy than even a 6-dimensional HMM. We provide further details about this experiment, including specifics of learning HQMMs, in the next chapter.

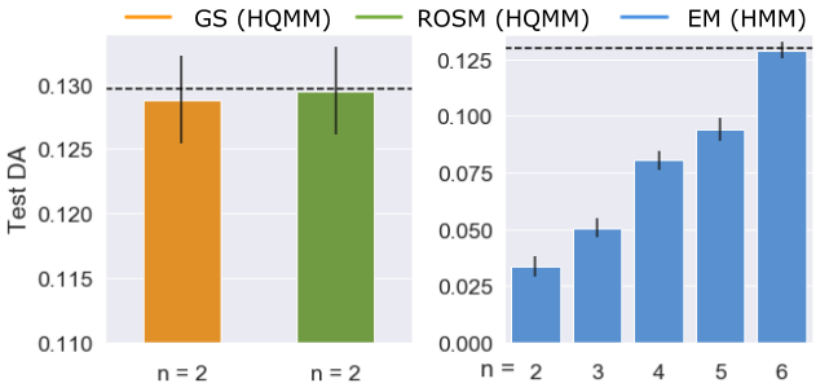


Figure 3.6: **Test Set Performance on the Synthetic HQMM Data**: The dashed line represents the test set performance of the true model that generated the data. The GS and ROSM methods were used to learn $(2, 6, 1)$-HQMMs, while EM was used to learn HMM models with varying number of hidden states (n). A $6-$state HMM model was needed to match a $2-$state HQMM. Full experimental details are provided in Section 4.4

Finally, we consider another aspect of compactness that is not captured in Definition 12 – compactness of the parameter space. Even if an HQMM is more compact than HMMs in terms of latent dimensions, it generally requires far more parameters. Specifically, an n-dimensional HMM with $s = |\mathcal{O}|$ observations is modeled via $n^2 + ns = n(n + s)$ learnable parameters. On the other hand, an n-dimensional HQMM requires $\sum_y^s (n \times n) \times w_y$. If we set the same number of operators $w_y = w$ for all observations (as we do in our experiments), we get wsn^2 learnable parameters. However, as we have discussed, this increase in parameter space can come with the additional benefit of increased expressiveness and potentially more compact latent space. Given the current scale of modern machine learning computations, the increase in parameter space will likely not

present a significant hurdle for most applications. However, being able to train such highly parameterized models requires efficient learning algorithms, which we now discuss in the next chapter.

3.8 Conclusion

In this chapter, we provided an introduction to HQMMs, illustrating how they are strictly more expressive than HMMs. In Chapters 5-7 we will establish additional relative expressiveness properties connecting HQMMs to other generalizations of HMMs, and establishing them as a highly expressive model class.

While expressiveness is a theoretically interesting concept, its practical significant hinges on being able to actually learn these models from data. The Stiefel manifold parameterization of HQMMs discussed in this chapter has been crucial in designing learning algorithms for HQMMs (Srinivasan et al., 2018c). In the next chapter, we discuss an existing approach that utilizes this Stiefel-parameterization. We highlight how this approach does not fully utilize the Riemannian parameterization, and propose an alternate approach that exploits the metric structure to learn HQMMs using gradient descent.

4
Learning HQMMs on the Stiefel Manifold

Abstract *Utilizing the geometric characterization of hidden quantum Markov model (HQMM) operators as elements of the Stiefel manifold, we propose training these models using Riemannian gradient descent. Prior to our approach, the only existing algorithm relied on learning HQMMs via iterated Givens rotations, which we refer to as Givens Search (GS) (Srinivasan et al., 2018e). We discuss how we overcome some of the shortcomings of GS, and experimentally demonstrate greater speed and modeling accuracy. Moreover, we show that our approach does not just provide incremental improvements, but, in fact, enables scaling up HQMMs to larger problem classes that were prohibitively slow to learn using GS.*

Hidden Quantum Markov Models (HQMMs) are known to be more expressiveness than the classical Hidden Markov Models (HMMs), but their applicability as a probabilistic graphical model for practical applications depends on being able to learn them from data. For HMMs, the Baum-Welch expectation maximization algorithm (Baum and Petrie, 1966) has been the algorithm for choice for decades. Devising a learning algorithm for HQMMs requires an efficient means of constraining learned operators to satisfy the requirements defined in Definition 11. As discussed in Section 3.6, all but the normalization constraints on Kraus operators are already satisfied by the form of the operator-sum representation – leaving the trace-preservation constraint as the only constraint that must be enforced explicitly.

In this chapter, we present an approach to learn valid HQMM operators utilizing its Stiefel manifold parameterization via Riemannian gradient descent introduced in Adhikary et al. (2020). Prior to this approach, the only available algorithm, to the best of our knowledge, was that introduced by Srinivasan et al. (2018d) that learns a sequence of Givens rotations to transform an initial set of HQMM parameters to ones that best describe the data. However, this approach, which we refer to as *Givens Search*, tends to be relatively slow for reasons we soon discuss. Coupled with the fact that the number of HQMM parameters increase much faster than HMM parame-

ters with both latent dimensions and number of observations, the limited speed of the Givens Search approach made it prohibitively expensive to experiment with large HQMMs. Thus, the application of HQMMs had been restricted to relatively simple settings that are not representative of real-world problems. We demonstrate how our approach that utilizes gradient descent adapted to the Riemannian Stiefel-manifold structure of HQMM allows us to mitigate this restriction, and apply it to problems that were previously impractical with Givens Search.

Outline We define the problem of learning HQMMs from data in Section 4.1, and summarize the Givens Search method that we use as a baseline – particularly focusing on its limitations. In Section 4.3, we discuss our approach of learning HQMMs using retractions on the Stiefel manifold (ROSM). In Section 4.4, we compare our approach against Givens Search on multiple synthetic datasets, including data from quantum processes, as well as a real-world dataset of DNA sequences. Through these experiments, we aim to demonstrate that ROSM provides higher accuracy and speed compared to Givens Search, and finally enables scaling to more complex real-world datasets.

Notation We use the same mathematical notation used in the earlier chapters as described in Section 3.1.

4.1 Learning HQMMs

The Learning Problem Consider a dataset of sequences $\mathcal{D} = \{\bar{y}_t\}$, where each sequence $\bar{y}_t = [y_{t_1}, \cdots, y_T]$ consists of T observations $y_t \in \mathcal{O}$. Following Srinivasan et al. (2018e), we can fit a model to this dataset by minimizing the following negative log-likelihood objective

$$\mathcal{L} = \sum_{\bar{y}_d \in \mathcal{D}} \mathcal{L}_{\bar{y}_d} = - \sum_{\bar{y}_d \in \mathcal{D}} \ln P(\bar{y}_d)$$

Given an HQMM $(\mathbb{C}^{n \times n}, \{\mathbf{K}_{y,w_y}\}_{y \in \mathcal{O}}, \hat{\boldsymbol{\rho}}_0, \mathrm{Tr}(\cdot))$ as defined in Definition 11, can we update the initial state $\hat{\boldsymbol{\rho}}_0$ conditioned on an initial observation y_0 as follows

$$\hat{\boldsymbol{\rho}}' = \sum_{w_{y_0}} \mathbf{K}_{y_0,w_{y_0}} \, \hat{\boldsymbol{\rho}}_0 \, \mathbf{K}_{y_0,w_{y_0}}^{\dagger}$$

Here $\hat{\boldsymbol{\rho}}'$ is the *unnormalized* updated latent state, and the likelihood of having observed y_0 is simply the sum of likelihoods over all latent states after the conditional update. This is simply the trace of the updated $\hat{\boldsymbol{\rho}}'$. To obtain the likelihood of a full sequence, we apply

the same update sequentially for every observation in the sequence, before ultimately applying the trace operation. The negative log-likelihood objective defined over a single sequence $\bar{y} = [y_0, \cdots, y_T]$ is thus computed as

$$\begin{aligned}\boldsymbol{\kappa}^* &= \operatorname*{argmin}_{\boldsymbol{\kappa}} \mathcal{L}_{\bar{y}}(\boldsymbol{\kappa}) \;\text{ s.t. }\; \boldsymbol{\kappa}^\dagger\boldsymbol{\kappa} = \sum_{w_y, y} \mathbf{K}^\dagger_{y,w_y}\mathbf{K}_{y,w_y} = \mathbb{I} \\ &= \operatorname*{argmin}_{\boldsymbol{\kappa}} -\ln \operatorname{Tr}\left(\sum_{w_{y_T}} \mathbf{K}_{y_T,w_{y_T}} \cdots \left(\sum_{w_{y_0}} \mathbf{K}_{y_0,w_{y_0}}\, \hat{\boldsymbol{\rho}}_0\, \mathbf{K}^\dagger_{y_0,w_{y_0}}\right) \cdots \mathbf{K}^\dagger_{y_T,w_{y_T}}\right) \\ &\qquad \text{s.t. }\; \boldsymbol{\kappa}^\dagger\boldsymbol{\kappa} = \sum_{w_y, y} \mathbf{K}^\dagger_{y,w_y}\mathbf{K}_{y,w_y} = \mathbb{I}\end{aligned} \tag{4.1}$$

Recall that $\boldsymbol{\kappa}$ is formed by vertically stacking all Kraus operators $\{\mathbf{K}_{y,w_y}\}$ across all observations, which, as denoted by the constraint, is an element of the Stiefel manifold. In the absence of this constraint, optimizing the objective would be fairly straightforward – we could simply update the parameters of $\boldsymbol{\kappa}$ using gradient descent. However, such updates would result in $\boldsymbol{\kappa}^*$ not being on the Stiefel manifold, and thus the learned model would not be a valid HQMM. The learned $\boldsymbol{\kappa}^*$ would then need to somehow be heuristically projected back onto the Stiefel manifold to ensure compliance. Since such heuristic projections generally do not account for the overall objective, they do not guarantee an expected decrease in the overall objective. Indeed, while the $\boldsymbol{\kappa}^*$ obtained via gradient descent will, in expectation, greedily reduce the objective, the intermediate heuristic projections break any guarantees – the objective value may even increase after the projection.

An alternative approach is to identify an optimization trajectory over the Stiefel manifold. In Adhikary et al. (2020), we proposed optimizing the above objective using Riemannian gradient descent, and present this approach in Section 4.3. But first, we present a approach that predated ours, and which, at the time, was the first and only available algorithm to learn HQMMs from data. We use this algorithm as our baseline, and show how our approach attempts to overcome its shortcomings.

4.2 Givens Search

To ensure that the Stiefel manifold constraint on $\boldsymbol{\kappa}$ is maintained, Srinivasan et al. (2018e) note that it is sufficient to enforce that updates made to $\boldsymbol{\kappa}$ are unitarity-invariant. Namely, given an initial $\boldsymbol{\kappa}$ that lies on the Stiefel manifold, all updates must continue to be on the Stiefel manifold. One option of such a Stiefel-invariant update is to restrict ourselves to rotations since rotations preserve unitarity. If

we pick a class of rotation transformations parameterized by some learnable parameters, we can then use the objective to learn these parameters. We could then turn the constrained optimization problem in Equation 4.1 into an unconstrained optimization problem through reparameterization.

Givens Rotations Srinivasan et al. (2018e) pick the family of complex Givens rotations as the class of parameterized unitary transformations. A complex Givens rotation matrix $\mathbf{H}(p, q, \theta, \phi, \psi, \delta)$ is simply the identity matrix $\mathbb{I}$ with just the elements corresponding to the p-th and q-th row and column replaced by elements parameterized by angles $(\theta, \phi, \psi, \delta)$ as follows:

$$
\begin{aligned}
\mathbf{H}_{k,k} &= 1 && \forall k \neq p, q \\
\mathbf{H}_{k,l} &= 0 && k \neq p, q \text{ and } l \neq p, q \\
\mathbf{H}_{k,k} &= \exp(i\phi/2)\exp(i\psi)\cos\theta && k = p \\
\mathbf{H}_{k,k} &= \exp(i\phi/2)\exp(-i\psi)\cos\theta && k = q \\
\mathbf{H}_{p,q} &= -\exp(i\phi/2)\exp(-i\delta)\sin\theta && p > q \\
\mathbf{H}_{p,q} &= \exp(i\phi/2)\exp(i\delta)\sin\theta && p < q
\end{aligned}
$$

Note that $\mathbf{H}(p, q, \theta, \phi, \psi, \delta)$ only affects the p-th and q-th rows of the matrix to which it is applied – all other elements undergo an identity transformation. In Figure 4.2, we provide a visual illustration of a $\mathbf{H}(1, 1, \theta, \phi, \psi, \delta)$.

Givens Search Having parameterized the updates for $\boldsymbol{\kappa}$, we no longer need to explicitly enforce the Stiefel constraints on $\boldsymbol{\kappa}$. We can thus initialize a complex-valued $\boldsymbol{\kappa}$ on the Stiefel manifold, and iteratively apply Givens rotations as follows:

$$\boldsymbol{\kappa}' = \mathbf{H}(p, q, \theta, \phi, \psi, \delta)\ \boldsymbol{\kappa}$$

However, we still do need to handle constraints on the parameters $(p, q, \theta, \phi, \psi, \delta)$. Firstly, given a tuple (p, q), the remaining parameters are all angles that are generally constrained within $[0, \pi]$. This is a fairly common constraint that is easily handled in most learning algorithms. As such, we can choose from a wide range of optimization algorithms; Srinivasan et al. (2018e) opt to use a gradient-free black-box optimization algorithm. The tricker constraint to satisfy is that both p and q must be integers. Srinivasan et al. (2018e) resolve this issue by picking random integers for (p, q), and then optimizing the remaining parameters conditioned on these integers. We illustrate the Givens Search algorithm in Figure 4.2.

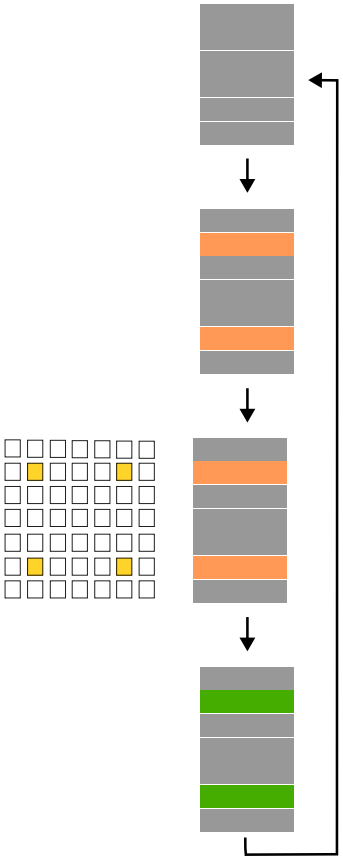


Figure 4.1: **The Givens Search Algorithm for HQMMs**: The Givens Search algorithm repeatedly performs the following iterative updates: (1) begin with a valid $\boldsymbol{\kappa}$ (2) randomly pick two rows (3) run a single iteration of an optimization algorithm to obtain a Givens rotation for the chosen rows, and (4) apply the rotation and obtain new $\boldsymbol{\kappa}'$. This is repeated until the objective converges, or a maximum number of iterations is reached.

Limitations of Givens Search We note that the above Givens Search (GS) approach has two crucial limitations. The first limitation is one of optimality – while the black-box optimization routine provides optimization guarantees on the learned angles $(\theta, \phi, \psi, \delta)$, but since indices (p, q) are picked at random, this optimization guarantee does not hold for $\boldsymbol{\kappa}$. The second limitation is one of speed – at every iteration, a Givens rotation only updates the i-th and j-th row of $\boldsymbol{\kappa}$. Thus, most of the parameters of $\boldsymbol{\kappa}$ are not optimized at every iteration. Recall that one of the drawbacks of HQMMs compared to HMMs is that the number of parameters for HQMMs scale quite rapidly compared to HMMs. Thus, the slowness of Givens Search quickly becomes a major hurdle as we increase the number of latent dimensions.

4.3 *Retractions on the Stiefel Manifold*

To overcome the limitations of Givens search highlighted in the earlier section, we propose directly learning $\boldsymbol{\kappa}$ using a gradient-based algorithm. 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 (Hjørungnes and Gesbert, 2007). We use the algorithm proposed by Wen and Yin (2013) to constrain our parameter updates on the Stiefel manifold.

The Wen-Yin Update Scheme Given a gradient $\mathbf{G}$ of the loss function $\mathcal{L}$ with respect to parameters $\boldsymbol{\kappa}$, we wish to find the trajectory $\gamma(\tau)$ for some step size τ that corresponds to stepping along the direction of the gradient while staying on the Stiefel manifold. The Wen-Yin approach achieves this through *retractions* that smoothly map $\mathbf{G}$ or any point on a manifold's tangent bundle onto the manifold itself, while preserving the descent direction at that point (Wen and Yin, 2013). We can intuitively think of a retraction as wrapping the direction of $\mathbf{G}$ onto the surface of the manifold. This provides us with a feasible path $\gamma(\tau)$ for curvilinear descent with respect to an initial feasible solution $\boldsymbol{\kappa}_0$:

$$\gamma(\tau) = \boldsymbol{\kappa}_0 - \tau \mathbf{U} \left(\mathbb{I} + \frac{\tau}{2} \mathbf{V}^\dagger \mathbf{U} \right)^{-1} \mathbf{V}^\dagger \boldsymbol{\kappa}_0, \tag{4.2}$$

where $\mathbf{U} = [\mathbf{G} \mid \boldsymbol{\kappa}_0]$, $\mathbf{V} = [\boldsymbol{\kappa}_0 \mid -\mathbf{G}]$, and $\mathbf{G}$ is the gradient at $\boldsymbol{\kappa}_0$. To see that $\gamma(\tau)$ is, in fact, the direction of steepest descent to feasibly optimize our loss, we can check if two important criteria are met. First, when $\tau = 0$, we should be at the initial point $\boldsymbol{\kappa}_0$ with $\gamma(0)$ pointing the same direction as $\mathbf{G}$. This is easily verified since $\gamma(0) = \boldsymbol{\kappa}_0$ and $\gamma'(0) = -\mathbf{G}$ (Wen and Yin, 2013). Second, any point along $\gamma(\tau)$ must be feasible. To confirm this, note that Equation 4.2 can be

equivalently written as the following Crank-Nicolson-like update

$$\gamma(\tau) = \left(\mathbb{I} + \frac{\tau}{2}\mathbf{A}\right)^{-1}\left(\mathbb{I} - \frac{\tau}{2}\mathbf{A}\right)\boldsymbol{\kappa}_0, \tag{4.3}$$

where $\mathbf{A} = \mathbf{G}\boldsymbol{\kappa}_0^\dagger - \boldsymbol{\kappa}_0\mathbf{G}^\dagger$. Thus, $\gamma(\tau)$ is actually the Cayley transform of the skew-symmetric matrix $\mathbf{A}$ applied to $\boldsymbol{\kappa}_0$. Using this interpretation, it can be shown (Wen and Yin, 2013) that $\gamma(\tau)^\dagger\gamma(\tau) = \boldsymbol{\kappa}_0^\dagger\boldsymbol{\kappa}$. Therefore, as long as the initial point is feasible ($\boldsymbol{\kappa}_0^\dagger\boldsymbol{\kappa}_0 = \mathbb{I}$), every point along $\gamma(\tau)$ will be feasible. While Equation 4.3 is easier to interpret, Equation 4.2 is computationally favorable as it requires the inversion of a smaller $2n \times 2n$ matrix.

We combine the Wen-Yin update with a simple gradient descent scheme (Algorithm 1) to learn feasible parameters for HQMMs. In our experiments with $N = |\mathcal{O}|w$, and for a batch with m sequences of length l, we compute the loss using Equation 4.1 in $O(mlwn^3)$ time, perform auto-differentiation, and obtain a retraction using Equation 4.2 in $O(|\mathcal{O}|wn^3)$ time.

Algorithm 1: Learning HQMMs with Retractions on the Stiefel Manifold

Input: Training data $\mathbf{Y} \in \mathbb{N}^{M\times\ell}$
Data: M: number of data points, ℓ: number of observed variables in HQMM
Parameters: τ: learning rate, α: learning rate decay, B: number of batches, E: number of epochs
Output: $\{\mathbf{K}_i\}_{i=1}^{|\mathcal{O}|w}$

Initialize complex orthonormal matrix on Stiefel manifold $\boldsymbol{\kappa} \in \mathbb{C}^{|\mathcal{O}|wn\times n}$;
Partition $\boldsymbol{\kappa}$ into Kraus operators $\{\mathbf{K}_i\}_{i=1}^{|\mathcal{O}|w}$, with each $\mathbf{K}_i \in \mathbb{C}^{n\times n}$;
for *epoch* $\leftarrow 1$ **to** E **do**
  Partition training data $\mathbf{Y}$ into B batches $\{\mathbf{Y}_b\}$;
  for $b \leftarrow 1$ **to** B **do**
    Compute gradients $\mathbf{G}_i \leftarrow \frac{\partial\mathcal{L}}{\partial\mathbf{K}_i^*}$ for batch $\mathbf{Y}_b$;
    Form $\mathbf{G} \leftarrow \begin{bmatrix}\mathbf{G}_1 & \cdots & \mathbf{G}_{|\mathcal{O}|w}\end{bmatrix}^T$;
    Construct $\mathbf{U} \leftarrow [\ \mathbf{G} \mid \boldsymbol{\kappa}\ ]$, $\mathbf{V} \leftarrow [\ \boldsymbol{\kappa} \mid -\mathbf{G}\ ]$;
    Update $\boldsymbol{\kappa} \leftarrow \boldsymbol{\kappa} - \tau\mathbf{U}\left(\mathbb{I} + \frac{\tau}{2}\mathbf{V}^\dagger\mathbf{U}\right)^{-1}\mathbf{V}^\dagger\boldsymbol{\kappa}$;
  Update learning rate: $\tau \leftarrow \alpha\tau$;
  Repartition $\boldsymbol{\kappa}$ into Kraus operators $\{\mathbf{K}_i\}$;
return $\{\mathbf{K}_i\}$

Revisiting Limitations of Givens Search Before delving into experiments, let us define our hypotheses for why we expect Algorithm 1

to outperform Givens Search by revisiting the two limitations of the latter discussed earlier. Firstly, unlike Givens Search, there is no need to resort to randomly selecting co-ordinate directions to optimize since the retraction based approach removes need for constrained integer optimization. Thus, we expect Algorithm 1 to find better optima than Givens Search, i.e., provide a better fit to the data.

Secondly, *all* parameters of $\boldsymbol{\kappa}$ get updated at every iteration, as opposed to the slow row updates in Givens Search. Thus, we expect Algorithm 1 to be much faster than Givens Search. We note that the slowness of the Givens Search algorithm is not just a minor inconvenience, but essentially makes it infeasible to learn large HQMMs with higher parameters. As we demonstrate, the improved speed provided by our approach allows us to finally experiment with large HQMM models, and begin to validate some of their theoretically-established properties.

4.4 Experiments

To show the superior performance of our apoorach using Retractions[1] on the Stiefel manifold (**ROSM**) over the previous Givens Search (**GS**) method in learning HQMMs, we evaluate their accuracy and run-time on three datasets. The first is the synthetic dataset used by Srinivasan et al. (2018e) that was generated by an HMM. The second is another synthetic dataset that models a simple quantum phenomena observed in the Stern-Gerlach experiment. Finally, we evaluate our algorithm on a real-world dataset, on which the GS approach is prohibitively slow; demonstrating the scalability of ROSM.

[1] In Adhikary et al. (2020), we had labeled this approach *Constrained* Optimization on the Stiefel Manifold (COSM). This was somewhat of a misnomer since retraction based algorithms are actually considered unconstrained optimization over manifolds. Here, we chose to rename the approach to better align with the literature.

Training For all our HQMMs, we use the log-likelihood loss function from Equation 4.1. We initialize the latent state $\hat{\boldsymbol{\rho}}_0$ as a random Hermitian PSD matrix using the QETLAB toolbox (Johnston, 2016), and $\boldsymbol{\kappa}$ as a random orthonormal matrix. Except for very small models, ROSM is fairly robust to random initializations (see Appendix 12.1). We compute the gradient of the loss function with respect to the complex conjugate of the Kraus operators using the Autograd package (which can handle complex differentiation), and vertically stack the gradients of the Kraus operators to construct the gradient $\mathbf{G}$ of the matrix $\boldsymbol{\kappa}$. To smoothen the trajectory we apply momentum with $\beta = 0.9$ (Rumelhart et al., 1986; Qian, 1999b), and re-normalize the gradient before and after the momentum update, making the magnitude of updates entirely dependent on step-size.

We refer to HQMMs using the tuple $(n, s, w)-$HQMM, where n is the number of hidden states, s is the number of possible outputs (earlier denoted $|\mathcal{O}|$), and w is the number of Kraus operators per output,

also referred to as the dimension of the 'environment' variable (Srinivasan et al., 2018e). Consequently, for an $(n, s, w)-$HQMM we have $\boldsymbol{\kappa} \in \mathbb{C}^{nsw \times n}$. We also provide the performance of HMMs trained using the Expectation-Maximization (**EM**) algorithm (with 5 random restarts) for reference. Details of our hyperparameter tuning procedure and computing infrastructure are described in Appendix 12.

Metrics In our experiments, we use a scaled log-likelihood (M. Zhao, 2007; Srinivasan et al., 2018e) independent of sequence length called description accuracy: $DA = f\left(1 + \frac{\log_s P(Y|\mathbb{D})}{\ell}\right)$, where $f(\cdot)$ squishes the log-likelihood from $(-\infty, 1]$ to $(-1, 1)$ (with $f(x) = \tanh(x/8)$ for $x \leq 0$, and $f(x) = x$ for $x > 0$). 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.

4.4.1 Synthetic HMM Data

For our first experiment, we generated data using the same synthetic HMM as Srinivasan et al. (2018e), with 6 hidden states and 6 outputs. We show two things with the experiments on this dataset: 1) ROSM finds better optima than GS, and 2) ROSM is much faster than GS – so much so that we could train larger HQMMs than were previously possible. We also investigate the effects of increasing model size by adding latent states (n) versus increasing the Kraus-rank (w).

We used the same 20 training and 10 validation sequences of length 3000 used by Srinivasan et al. (2018e), splitting up each sequence into 300 sequences and use a burn-in of 100. We trained HQMMs using the ROSM 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 4.2.

ROSM finds better optima than GS As shown in Figure 4.2, HQMMs (with $w = 1$) learned using ROSM achieve better optima than HQMMs learned using GS for all n. We also confirm that as noted in Srinivasan et al. (2018e), small HQMMs ($n \leq 5$) can model this data better than small HMMs, although this doesn't hold for $n = 6$. However, we can take advantage of the additional Kraus-rank hyperparameter w available to HQMMs to further improve performance, as shown in Figure 4.2 for $(5, 6, w)-$HQMMs (varying w). Moreover, note that the number of parameters for an HQMM scales faster than for an HMM.

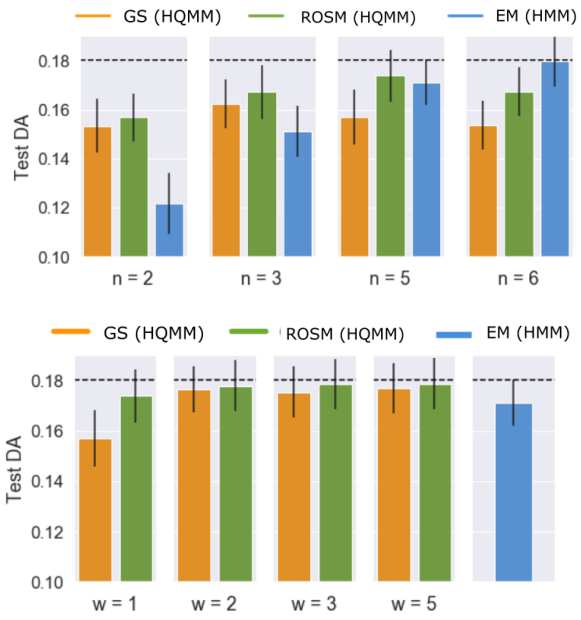


Figure 4.2: **Test Set Performances on the Synthetic HMM Data**: The dashed line represents the test set performance of the true model (a (6,6)-HMM) that generated the data.

ROSM is much faster than GS In Figure 4.3, 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).

For all models, we see that ROSM converges much faster than GS, and the difference in both speed and accuracy is especially pronounced for the larger models; ROSM quickly approaches convergence 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 did not directly calculate a precise speedup but provide an estimate in Appendix 12.3.

Srinivasan et al. (2018e) proved 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 ROSM, 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 sufficient to model our $(6,6)-$HMM.

4.4.2 *Experiment on Synthetic HQMM Data*

As an additional experiment on a purely quantum mechanical dataset, we compared the ROSM and GS methods on data generated using the synthetic HQMM with 2 hidden states and 6 possible outputs in Srinivasan et al. (2018e). The data generation process is

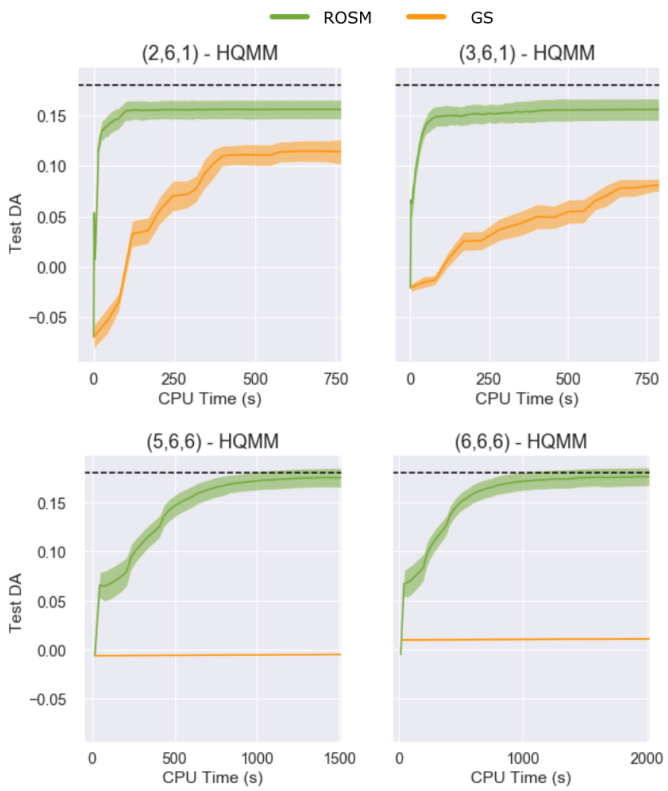


Figure 4.3: **ROSM Learns More Accurate Models Faster than GS**: Test DA versus training time for various (n, s, w)-HQMMs trained on the synthetic HMM data. ROSM converges to a better optimum faster than GS for all models; the dashed line represents the DA of the true data generating model.

inspired by the well known Stern-Gerlach experiment (Gerlach and Stern, 1922) in quantum mechanics, and at least 4 hidden states are required to model it. Srinivasan et al. (2018e) demonstrated that HQMMs *learned* from such synthetic data showed in practice the same benefits that held in theory. Our goal is to verify that the ROSM method performs at least as well as the GS method on a dataset well-suited to the HQMM model class.

We used the same synthetic dataset used by Srinivasan et al. (2018e), with 20 training and 10 validation sequences of length 3000. We further split up each sequence into 300 sequences and use a burn-in of 100, instead of training on 3000-length sequences with a burn-in of 1000. This reduced training time without impacting accuracy or the amount of training data processed. We trained HQMMs using the ROSM approach for 60 epochs, and saved the model that yielded the highest DA score on the validation set; we used this model to evaluate on the test set of 10 sequences of length 3000 (with burn-in 1000). The results for this model are shown in Figure 4.4.2. We see that the ROSM method achieves slightly better DA compared to the GS method. We confirm that as seen in Srinivasan et al. (2018e), we need a 6−state HMM to model this 2−state HQMM.

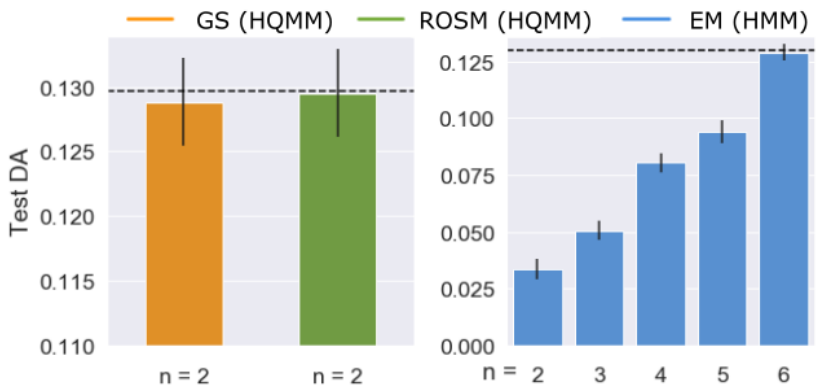


Figure 4.4: **Test Set Performance on the Synthetic HQMM Data**: The dashed line represents the test set performance of the true model that generated the data. The GS and ROSM methods were used to learn $(2, 6, 1)$-HQMMs, while EM was used to learn HMM models with varying number of hidden states (n). A $6-$state HMM model was needed to match a $2-$state HQMM.

4.4.3 *Splice Dataset*

For our final 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 one of four nucleobases: 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. We demonstrate that ROSM can be used to train HQMMs on real-world datasets which would have been too slow to train using GS.

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 ROSM method and HMMs with the EM algorithm (with 5 random restarts) for reference. In Figure 4.5, 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.

Note that 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 ROSM. 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. However unlike the previous experiment, we still see persistent gains by increasing n. Interpreting this in conjunction with the results

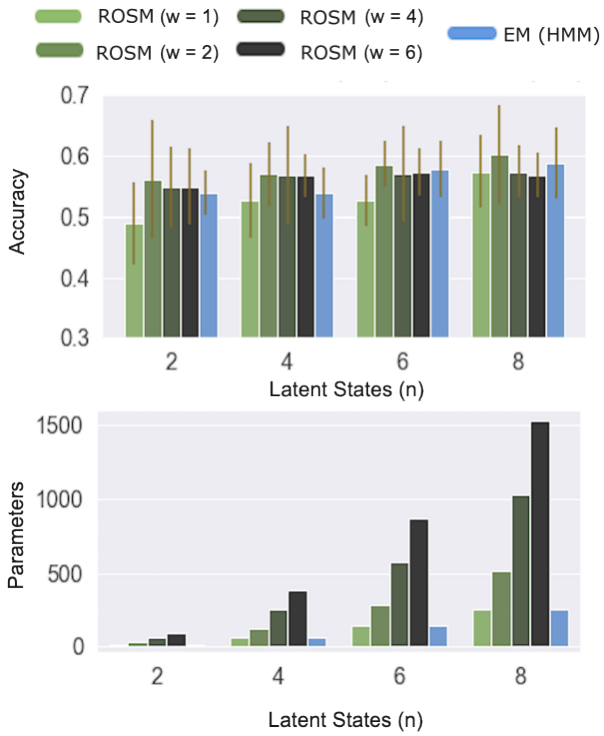


Figure 4.5: **Average 5-fold Test Set Performance on the Splice Dataset** Test set accuracies (top) and number of parameters (bottom) for various HQMMs and HMMs trained using the ROSM and EM algorithms respectively. Error bars in the top graph represent the mean standard deviation across labels over the 5 folds.

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, ROSM 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.

Alternative Update Schemes on the Stiefel Manifold As discussed in Section 4.3, we used the Wen-Yin update scheme to obtain retractions on the Stiefel manifold. This update is one of many options we could have chosen to define our updates. We now discuss some of these alternatives.

Algorithms that restrict parameters on the Stiefel manifold are generally 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 updates, those proposed by Wen and Yin (2013) and Jiang and Dai (2013) were (at least at the time of our proposal) 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 $O(7Nn^2)$, where n is the latent dimension and $N = |\mathcal{O}|w$. By comparison, the canonical gradient projection algorithm has a slightly lower computational complexity of $O(3Nn^2)$.

The exact update schemes and complexity calculations for all three methods can be found in Jiang and Dai (2013).

We compared these three update schemes by training multiple HQMMs for both synthetic HQMM and HMM datasets. We present these results in Appendix 12.4, and note here that all three methods yielded relatively similar speed and accuracy on 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.

4.5 Conclusion

By leveraging the Stiefel manifold parameterization of HQMMs, we proposed an efficient learning method for HQMMs based on Riemannian gradient descent. This approach addresses key limitations of the earlier Givens Search method by Srinivasan et al. (2018d), which relied on a sequential composition of rotations and proved to be computationally expensive – particularly for large HQMMs. Given that the number of parameters in HQMMs grows rapidly with both the latent state dimension and observation space, this inefficiency significantly constrained prior applications to only small examples. By contrast, our method enables practical training of larger HQMMs, thereby overcoming prior scalability barriers and making it feasible to explore HQMMs in more complex problem domains.

5
Expressiveness of HQMMs

Abstract *We expand our analysis of the expressiveness of HQMMs by comparing them to other generalizations of HMMs from classical stochastic process theory. We find that the 'quantum-inspired' modifications introduced in HQMMs actually fit neatly into a spectrum of models commonly known as observable operator models (OOMs) (Jaeger, 2000) or predictive state representations (PSRs) (Singh et al., 2004). By formulating the quantum-inspired HQMM as a sub-class of classical OOMs, we discuss how the increased expressiveness of HQMMs is driven by the relaxation of non-negativity requirements for states and operators. Moreover, the specific metric structure enforced on HQMM parameters, allows these models to circumvent a critical problem with general OOMs – the negative probability problem.*

Proposed as quantum-inspired generalizations of Hidden Markov Models (HMMs), Hidden Quantum Markov Models (HQMMs) (Monras et al., 2010) were shown to have greater expressiveness as they can model certain processes with finite dimensions that would otherwise require infinite-dimensional HMMs. Equipped with the learning algorithm introduced in Chapter 4, we are also able to learn them efficiently from data – sometimes providing a more compact model with fewer latent dimensions than HMMs.

Having established the greater expressiveness of HQMMs over HMMs, a few natural question arise: what exactly is driving this increased expressiveness? Is there some innate advantage of using quantum probabilistic dynamics that cannot be obtained through classical probability theory? Are there other model classes that also offer similar expressiveness? In this chapter, we begin to answer these questions by expanding the comparison of HQMMs to alternative classical (non-quantum) generalizations of HMMs known as Observable Operator Models (OOMs) (Jaeger, 2000) or Predictive State Representations (PSRs) (Singh et al., 2004).

As we discuss, the key to increased expressiveness of HQMMs is relaxing non-negativity restrictions on states and operators. Although we motivated these relaxations as intrinsic properties of quantum operations underlying HQMMs in Chapter 3, they are not exclusive

to quantum channels. Indeed, similar ideas have been adopted in the literature of classical stochastic process. Specifically, OOMs/PSRs, which have no quantum-mechanical probabilistic underpinnings, form the most expressive model class for linear stochastic processes. The relaxation of non-negativity constraints on HQMM states and parameters make them more expressive than HMMs, and some other model classes. On the other hand, its PSD/Stiefel metric constraints limit its expressiveness, but, as we detail, they also provide some necessary structure that make them well suited for learning from data.

Outline We begin with some background on linear stochastic processes in Section 5.1, and in Section 5.2 discuss the observable operator model, a general class of sequential system models that can capture such processes. These models have also been essentially introduced in slightly different context and names as predictive state representations (PSRs) and stochastic weighted automata (SWA). In Section 5.3 we describe the norm observable operator model (NOOM) that attempts to circumvent the negative probability problem, but, as we show, are not expressive enough to capture all finite HMMs. In Section 5.4, we introduce a new formulation HQMMs as a generalization of NOOMs, establishing HQMMs as OOMs. We demonstrate that HQMMs contain both NOOMs and HMMs as subclasses, and are thus an attractive model class with relatively high expressiveness but sufficient constraints to avoid the negative probability problem.

Notation We use the same mathematical notation used in the earlier Chapters as described in Section 3.1.

5.1 Sequential Systems and Linear Stochastic Processes

In our analysis of HQMMs and related models, we are interested in models that capture stochastic processes that produce sequences of observations. Particularly, both HQMMs and HMMs model the conditional probabilities of such sequences through linear operations. To contextualize these models with broader model classes, we turn to the general theory of sequential systems (Schützenberger, 1961; Carlyle and Paz, 1971; Fliess, 1974) – particularly their linear variants which as eventually show, include HQMMs.

Following Thon and Jaeger (2015), we define a sequential system as follows:

Definition 13 (Sequential Systems (Thon and Jaeger, 2015))**.** *An n-dimensional sequential system*[1] *with a set of discrete observations $\mathcal{O}$ is*

[1] Sequential systems can, in general, be defined over arbitrary scalar fields $\mathbb{K}$, i.e., $\boldsymbol{\omega}_0 \in \mathbb{K}^d, \boldsymbol{\tau} \in \mathbb{K}^{d\times d}, \sigma : \mathbb{K}^d \to \mathbb{K}$. We have set $\mathbb{K} = \mathbb{C}$ for simplicity, as it is covers all models we analyze.

defined as a tuple $\mathcal{S} = (\sigma, \{\boldsymbol{\tau}_y\}, \boldsymbol{\omega}_0)$ *consisting of an initial state vector* $\boldsymbol{\omega}_0 \in \mathbb{C}^n$*, a matrix* $\boldsymbol{\tau}_y \in \mathbb{C}^{n \times n}$ *for each* $y \in \mathcal{O}$*, and an evaluation functional* $\sigma : \mathbb{C}^n \to \mathbb{C}$*.*

Given a sequence of observations $\bar{y} = y_0 \cdots y_T \in \mathcal{O}^{\otimes T}$*, let* $\boldsymbol{\tau}_{\bar{y}} = \boldsymbol{\tau}_{y_T} \cdots \boldsymbol{\tau}_{y_0}$*. The state of the sequential system conditioned on* $\bar{y}$ *is computed as*

$$\boldsymbol{\omega}_{\bar{y}} = \boldsymbol{\tau}_{y_T} \cdots \boldsymbol{\tau}_1 \boldsymbol{\omega}_0,$$

and the evaluation function $f_{\mathcal{S}}$ *of the system is defined as follows*

$$f_{\mathcal{S}}(\bar{y}) = \sigma \boldsymbol{\tau}_{\bar{y}} \boldsymbol{\omega}_0$$

Note that while the evaluation function $f_{\mathcal{S}}(\bar{y})$ could be used to model any arbitrary scalar summary of the sequence $\bar{y}$, it generally corresponds to the probability $P(\bar{y})$ of observing the sequence.

Within the general class of sequential systems, we will focus on a particular sub-class of *linear* sequential systems:

Definition 14 (Linear Sequential System)**.** *A d-dimensional linear sequential system is a sequential system where the the evaluation functional* σ *corresponds to the inner product with respect to a vector* $\boldsymbol{\sigma} \in \mathbb{C}^d$

$$f_{\mathcal{S}} = \sigma \boldsymbol{\tau}_{\bar{y}} \boldsymbol{\omega}_0 = \boldsymbol{\sigma}^\dagger \boldsymbol{\tau}_{\bar{y}} \boldsymbol{\omega}_0$$

To identify what types of processes can be modeled as a linear sequential system, we can define a useful construct known as the *Hankel* matrix $\mathbf{H}$ (Jaeger, 2000; Balle et al., 2014b; Thon and Jaeger, 2015), also known as the *system-dynamics* matrix (Singh et al., 2004). The elements of the Hankel matrix correspond to the probability of observing every possible future sequence of observations $\bar{y}_f$ conditioned on having observed a past sequence of observations $\bar{y}_p$, where both the sequences can be of arbitrary length. Specifically, imagine listing out all possible arbitrary length sequences and enumerating them from $[0, \infty]$. The j-th column of the Hankel matrix, $\mathbf{H}[j, :]$ then lists out the conditional probability of observing the (future) sequence $y_f^{(j)}$ conditioned on all other possible sequences of past observations y_p:

$$\mathbf{H}[:, j]^T = [P(y_f^j | y_p^{(0)}), P(y_f^j | y_p^{(1)}), s \cdots, \cdots]^T$$

Similarly, the i-th row of $\mathbf{H}$ lists the probability of observing all possible future observation sequences y_f conditioned on having observed the past sequence $y_p^{(i)}$

$$\mathbf{H}[i, :] = [P(y_f^0 | y_p^{(i)}), P(y_f^1 | y_p^{(i)}), s \cdots, \cdots]$$

The Hankel matrix $\mathbf{H}$ is thus a bi-infinite matrix with infinite rows and columns, and as such, is not a particularly useful tool for computations. However, if $\mathbf{H}$ has a finite rank r, then we can construct any column of $\mathbf{H}$ using a weighted combination of r linearly independent columns of $\mathbf{H}$. Since the Hankel matrix fully characterizes a stochastic process by listing all possible conditional probabilities of observations, these r linearly independent columns are sufficient to predict the probability of any future sequence conditioned on any past sequence. The rank of the Hankel matrix of a stochastic process, or sometimes called just the rank of the stochastic process, is directly related to whether it can be modeled by a linear stochastic sequential system

Remark 4 ((Thon and Jaeger, 2015)). *If the Hankel matrix of a stochastic process has finite rank $d < \infty$, the process can be exactly modeled by a d-dimensional linear sequential system.*

We thus obtain the general class of stochastic processes that can be modeled by linear sequential systems for which arbitrary columns of the Hankel matrix – descriptions of arbitrary futures conditioned on all possible pasts – can be formed via linear combinations of linearly independent columns. Specifically, these processes are defined as follows:

Definition 15 (Linear Stochastic Process). *A stochastic process is linear if it is Hankel matrix has finite rank $d < \infty$. These processes can be modeled exactly by d-dimensional linear stochastic systems.*

We now discuss a class of finite-dimensional sequential systems called observable operator models that can model all linear stochastic processes.

5.2 *Observable Operator Models (OOMs)*

Observable operator models were originally introduced by Jaeger (1998) as general models of linear stochastic processes (Jaeger, 2000; Thon and Jaeger, 2015). We present this original formulation of OOMs as standard OOMs as follows

Definition 16 (Standard OOMs (Jaeger, 2000)). *An n-dimensional standard Observable Operator Model with a set of discrete observations $\mathcal{O}$ is a tuple $(\mathbb{R}^n, \{\mathbf{T}_y\}_{y\in\mathcal{O}}, \mathbf{x}_0)$ where initial state $\mathbf{x}_0 \in \mathbb{R}^n$ and observable operators $\{\mathbf{T}_y\}_{y\in\mathcal{O}} \in \mathbb{R}^{n\times n}$ satisfy the following constraints:*

(i) Normalized initial state: $\vec{1}^T \mathbf{x}_0 = 1$,

(ii) Normalized marginal over observations: $\vec{1}^T \sum_{y\in\mathcal{O}} \mathbf{T}_y = \vec{1}^T$,

(iii) Non-negative probabilities: $\vec{\mathbb{1}}^T \mathbf{T}_{y_t} \dots \mathbf{T}_{y_1} \mathbf{x}_0 \geq 0$ *for all sequences* $y_1 \dots y_t$.

General OOMs The standard OOMs given in Definition 16 follow the original formulation by Jaeger (2000), which is stricter than necessary. Various equivalent formulations have been proposed, including quasi-realizations (Vidyasagar, 2011), (uncontrolled) predictive state representations (PSRs), and stochastic weighted automata (SWA) (Balle et al., 2014b). Developed from the perspective of sequential decision making, PSRs share much of the same intuitions behind finite-rank Hankel matrices discussed in Section 5.1. Singh et al. (2004) refer to the rows of these independent columns of the Hankel matrix as *predictive states* as they are sufficient statistics to predict the future. Since these models were developed within the context of controlled processes, PSRs are generally defined with additional control parameters – in this chapter, we restrict ourselves to uncontrolled process, and discuss models with control in Chapter 6.

Similar to PSRs, SWAs were developed with the same ideas as OOMs, but from the formal language community to model probability distributions over sequences of symbols drawn from an alphabet describing a formal language. It is worth mentioning that the semantics of the probabilities computed by PSRs and stochastic WAs can differ: while PSR typically maintain a recursive state and are used to compute the probability of a given sequence conditioned on some past sequence, stochastic WA are often used to compute the joint distributions over the set of all possible finite length sequences. We refer the reader to Thon and Jaeger (2015) for a unifying perspective discussing these and other related models.

We consider OOMs, PSRs and SWAs collectively as instances of 'general OOMs'. The main difference is that the model parameters are no longer constrained to be real, and we do not force the state entries to sum to one; instead the state can be any vector as long as we can use a linear functional $\boldsymbol{\sigma}$ (which for standard OOMs was fixed to be $\vec{\mathbb{1}}^T$) to recover the probabilities. While the model parameters can be defined over arbitrary fields, we define general OOMs over the complex field as this allows us to eventually recover HQMMs.

Definition 17 (General OOMs (Thon and Jaeger, 2015))**.** *An n-dimensional general Observable Operator Model with a set of discrete observations* $\mathcal{O}$ *is a tuple* $(\mathbb{C}^n, (\boldsymbol{\tau}_y)_{y\in\mathcal{O}}, \mathbf{x}_0, \boldsymbol{\sigma})$ *where initial state* $\mathbf{x}_0 \in \mathbb{C}^n$, *observable operators* $\{\boldsymbol{\tau}_y\}_{y\in\mathcal{O}} \in \mathbb{C}^{n\times n}$, *and a linear evaluation functional* $\boldsymbol{\sigma} \in \mathbb{C}^n$ *satisfy the following constraints:*

(i) Normalized Initial State: $\boldsymbol{\sigma}^\dagger \mathbf{x}_0 = 1$,

(ii) Normalized marginal over observations: $\boldsymbol{\sigma}^\dagger \boldsymbol{\tau}_{y_t} \dots \boldsymbol{\tau}_{y_1} x_0 = \sum_{y \in \mathcal{O}} \boldsymbol{\sigma}^\dagger \boldsymbol{\tau}_y \boldsymbol{\tau}_{y_t} \dots \boldsymbol{\tau}_{y_1} \mathbf{x}_0$ *for all sequences* $y_1 \dots y_t$,

(iii) Non-negative probabilities: $\boldsymbol{\sigma}^\dagger \boldsymbol{\tau}_{y_t} \dots \boldsymbol{\tau}_{y_1} \mathbf{x}_0 \in [0, 1]$ *for all sequences* $y_1 \dots y_t$.

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

$$\mathbf{x}_t = \frac{\boldsymbol{\tau}_y \mathbf{x}_{t-1}}{\boldsymbol{\sigma}^\dagger \boldsymbol{\tau}_y \mathbf{x}_{t-1}} \qquad P(\bar{y}) = \boldsymbol{\sigma}^\dagger \boldsymbol{\tau}_{y_t} \cdots \boldsymbol{\tau}_{y_1} \mathbf{x} \tag{5.1}$$

As shown in Proposition 13 of Thon and Jaeger (2015), every n-dimensional general OOM has an equivalent standard OOM that is a similarity transform away, i.e., we can find a similarity transform $\mathbf{S}$ such that $(\mathbb{C}^n, (\mathbf{S}\boldsymbol{\tau}_y\mathbf{S}^{-1})_{y\in\mathcal{O}}, \mathbf{S}\boldsymbol{\omega}_0, \boldsymbol{\sigma}\mathbf{S}^{-1}) = (\mathbb{C}^n, (\mathbf{T}_y)_{y\in\mathcal{O}}, \mathbf{v}_0, \vec{\mathbb{1}}^T)$. Finally, we recall that that finite dimensional OOMs are the most expressive class of linear models capable of modeling any linear stochastic process whose 'system-dynamics' matrix (Singh et al., 2004) has finite rank (Zhao and Jaeger, 2010b). Hence these models are extremely powerful.

State Space Geometry A useful conceptual characterization of a candidate OOM with parameters $\{\mathbf{T}_y\}_{y\in\mathcal{O}}$ is the convex cone of valid initial states it admits, i.e., the initial states for which the model will never assign a negative probability for observations. If there is no such cone, the model is invalid. Indeed, Jaeger (2000) present the following alternative to condition (iii):

Proposition 1 (Jaeger (2000))**.** *A tuple* $(\mathbb{R}^n, (\mathbf{T}_y)_{y\in\mathcal{O}}, \mathbf{x}_0)$ *satisfying conditions (i)-(ii) of Definition 16 is an OOM if and only if there exists a pointed convex cone* $\mathcal{K}$ *such that:*

(i) Initial state is in the cone: $\mathbf{x}_0 \in \mathcal{K}$,

(ii) $\mathcal{K}$ *is closed under the operators:* $\mathbf{T}_y\mathbf{x} \in \mathcal{K}$ *for all* $\mathbf{x} \in \mathcal{K}$ *and* $y \in \mathcal{O}$,

(iii) The sum of entries for any point in the cone is non-negative: $\vec{\mathbb{1}}^T\mathbf{x} \geq 0$ *for all* $\mathbf{x} \in \mathcal{K}$.

Conditions (i) and (ii) guarantee that any initial state inside such a cone will stay inside the cone under action of $\mathbf{T}_y$, and condition (iii) guarantees that any state inside the cone will evaluate to a non-negative probability.

The Negative Probability Problem (NPP) Note that the definition of OOMs is non-constructive since it does not tell us what constraints we could place on model parameters or initial states to satisfy condition (iii) – this is the cost of relaxing the non-negativity constraint.

In fact, it is *undecidable* whether a given candidate OOM $(\mathbb{R}^n, (\mathbf{T}_y)_{y\in\mathcal{O}}, \mathbf{x}_0)$ satisfying conditions (i)-(ii) will violate condition (iii) (Wiewiora, 2007). This is the root of the infamous negative probability problem (NPP) in OOMs, since we cannot identify whether a learned model will assign negative probabilities to observations. HMMs avoid the NPP through non-negativity constraints on states and operators but, as we now discuss, this restricts their expressiveness.

HMM $\subset$ OOM The familiar definition of Hidden Markov Models (HMMs) in Definition 10 can be reformulated as an OOM

Definition 18. *Given an n-dimensional Hidden Markov Model* $(\mathbb{R}^n, \mathbf{A}, \mathbf{C}, \mathbf{x}_0)$ *as defined in Definition 10, its observable operator* $\mathbf{T}_y$ *corresponding to observing y is defined as* $\mathbf{T}_y = \texttt{diag}(\mathbf{C}_{(y,:)})\mathbf{A}$*, where the matrix* $\mathbf{C}_{(y,:)}$ *is a diagonal matrix formed from the y-th row of* $\mathbf{C}$*. The state update is done as* $\mathbf{x}_t = \frac{\mathbf{T}_y\ \mathbf{x}_{t-1}}{\vec{\mathbf{1}}^T\mathbf{T}_y\ \mathbf{x}_{t-1}}$*, and the probability of observing sequences of observations is computed as* $P(y_1, \ldots, y_N) = \vec{\mathbf{1}}^T\mathbf{T}_{y_N}\ldots\mathbf{T}_{y_1}\mathbf{x}$*;*

Jaeger (2000) showed that HMM $\subset$ OOM using the 'probability clock' OOM which requires an infinite-dimensional HMM to model. Jaeger (2000) note that the non-negativity constraint for HMMs (Property (i) from Definition 18) forces the largest eigenvalue of observable operators $\mathbf{T}_y$ of an HMM to be real (by the Perron-Frobenius theorem). However, negative entries in OOMs allow the largest eigenvalue to be complex, which allows the latent states (and hence conditional probabilities) to display oscillatory behaviour. Jaeger (2000) uses this property in their probability clock example.

Since HMMs are decidedly less expressive than OOMs, it may be surprising that HMMs remain ubiquitous across multiple applications requiring sequential probabilistic modeling. Why not just swap HMMs with the more expressive OOMs? Besides the availability of efficient learning algorithms, the NPP is a fundamental reason one might opt for an HMM over OOMs.

Since the NPP is a strict undecidability result, it is not possible to devise an algorithm that can learn a general unconstrained OOM that will reliably only return non-negative probabilities without resorting to some heuristics. On the contrary, despite its limited expressiveness, HMMs do not suffer from the NPP and thus enable the use of non-heuristic algorithms with strong guarantees. This prompts a natural line of research – can we devise alternatives to HMMs that are more expressive, but still avoid the NPP? We now discuss one such proposal: Norm Observable Operator Models (NOOMs). We will eventually show that HQMMs can be formulated as general OOMs

with additional constraints to avoid the NPP, by formulating them as generalization of NOOMs.

5.3 Norm Observable Operator Models (NOOMs)

The convex-cone-characterization of OOMs provides a recipe to develop sub-classes of OOMs that avoid the NPP: select a desired convex cone of valid initial states and construct operators such that the cone is closed under their action. The norm observable operator models introduced by Zhao and Jaeger (2010a) can be viewed as doing exactly that. Specifically, these models are defined as follows:

Definition 19 (NOOMs (Zhao and Jaeger, 2010b))**.** *An n-dimensional Norm Observable Operator Model with a set of discrete observations $\mathcal{O}$ is a tuple $(\mathbb{R}^n, (\boldsymbol{\phi}_y)_{y\in\mathcal{O}}, \mathbf{v}_0)$ where initial state $\mathbf{v}_0 \in \mathbb{R}^n$ and observable operators $\{\boldsymbol{\phi}_y\}_{y\in\mathcal{O}} \in \mathbb{R}^{n\times n}$ satisfy the following constraints:*

(i) Normalized initial state: $\|\mathbf{v}_0\|_2^2 = 1$,

(ii) Normalized marginal over observations: $\sum_{y\in\mathcal{O}} \boldsymbol{\phi}_y^\dagger \boldsymbol{\phi}_y = \mathbb{I}$.

The updated state after observing $y \in \mathcal{O}$ and the probability of that observation can be computed as

$$\mathbf{v}_t = \frac{\boldsymbol{\phi}_y \mathbf{v}_{t-1}}{\|\boldsymbol{\phi}_{y_t} \ \dots \ \boldsymbol{\phi}_{y_1}\mathbf{v}\|} \qquad P(\bar{y}) = \|\boldsymbol{\phi}_{y_t} \ \dots \ \boldsymbol{\phi}_{y_1}\mathbf{v}\|^2 \tag{5.2}$$

Expressiveness of NOOMs Although any linear stochastic process can be represented as a NOOM in some inner product space, this space may be infinite dimensional (Zhao and Jaeger, 2010b). For practical purposes, we care about the expressiveness of finite-dimensional NOOMs. Zhao and Jaeger (2010b) showed that NOOM $\subseteq$ OOM, and once again used the ability of a real-valued NOOM operator to have complex eigenvalues in a NOOM probability clock to show that there are finite-dimensional NOOMs that cannot be modeled exactly by finite-dimensional HMMs.

Zhao and Jaeger (2010b) show that NOOMs are equivalent to n^2-dimensional OOMs, and indeed we will build upon this approach to re-derive HQMMs. Zhao and Jaeger (2010b) use Kronecker product relationships for the 2-norm (where $\vec{\mathbb{I}}$ is a vectorized identity matrix that implements a matrix trace operation) to show that sequence probabilities in a NOOM from Equation 5.2 can also be evaluated as:

$$P(\bar{y}) = \vec{\mathbb{I}}_{n^2}^T \ (\boldsymbol{\phi}_{y_t} \otimes \boldsymbol{\phi}_{y_t}) \ \dots \ (\boldsymbol{\phi}_{y_1} \otimes \boldsymbol{\phi}_{y_1}) \, (\mathbf{v}_0 \otimes \mathbf{v}_0) \,, \tag{5.3}$$

Now, if we define $\boldsymbol{\sigma} = \vec{\mathbb{I}}_{n^2}$, $\boldsymbol{\tau}_y = \boldsymbol{\phi}_y \otimes \boldsymbol{\phi}_y$, and the initial state $\boldsymbol{\omega}_0 \in \mathbb{R}^{n^2}$ as $\boldsymbol{\omega}_0 = \mathbf{v}_0 \otimes \mathbf{v}_0$, we get a general OOM $(\mathbb{C}^n, (\boldsymbol{\tau}_y)_{y\in\mathcal{O}}, \boldsymbol{\omega}_0, \boldsymbol{\sigma})$. As

shown by Zhao and Jaeger (2010b), this is a similarity transform of a standard OOM, with $\mathbf{S} = \mathbb{I}_{n^2} + \frac{1}{n^2}\vec{\mathbf{1}}_{n^2}(\boldsymbol{\sigma}^T - \vec{\mathbf{1}}_{n^2}^T)$. Thus, NOOMs are not any more expressive than OOMs, i.e., NOOM $\subseteq$ OOM.

While NOOMs manage to avoid the NPP without enforcing non-negative parameters, it is unclear if this necessarily provides greater expressiveness than HMMs. Zhao and Jaeger (2010b) have shown that finite HMMs cannot model all finite NOOMs (NOOM $\not\subseteq$ HMM), by constructing a NOOM that can model the probability clock (Figure 3.5), which cannot be modeled exactly by a finite HMM. However, the authors leave open the question of whether NOOMs subsume HMMs. In Adhikary et al. (2021a), we show that this is not the case

Theorem 5 (HMM $\not\subseteq$ NOOM)**.** *There exist finite-dimensional hidden Markov models that have no equivalent finite-dimensional norm-observable operator model.*

The proof relies on the fact that any two equivalent PSRs are related by a similarity transform (Thon and Jaeger, 2015). However, we find that any such similarity transform between HMMs and NOOMs will violate the normalization requirement of NOOM states.
We first introduce a lemma (Ito et al., 1992; Vidyasagar, 2011; Thon and Jaeger, 2015) that will help us in our proof. It tells us that two equivalent PSRs of the same dimension are simply a similarity transform away from one another.

Lemma 6 (Thon and Jaeger (2015))**.** *Suppose* $(\mathbb{C}^n, \boldsymbol{\sigma}, \{\boldsymbol{\tau}_y\}_{y\in\mathcal{O}}\}, \mathbf{x}_0)$ *and* $(\mathbb{C}^n, \boldsymbol{\sigma}', \{\boldsymbol{\tau}'_y\}_{y\in\mathcal{O}}\}, \mathbf{x}'_0)$ *are two equivalent PSR representations, i.e., they generate the same sequence of probabilities. Then, there exists some non-singular* $\mathbf{S} \in \mathbb{C}^{n\times n}$ *such that* $\mathbf{x}'_0 = \mathbf{S}^{-1}\mathbf{x}_0$, $\boldsymbol{\tau}'_y = \mathbf{S}^{-1}\boldsymbol{\tau}_y\mathbf{S}$, *and* $\boldsymbol{\sigma}'^T = \boldsymbol{\sigma}^T\mathbf{S}$.

Proof. We construct a class of HMMs for which there are no equivalent NOOMs and give a proof by contradiction. Let $\mathcal{A} = (\mathbb{R}^p, \boldsymbol{\sigma}, \{\boldsymbol{\tau}_y\}_{y\in\mathcal{O}}, \mathbf{x}_0)$ 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., $\mathbf{x}_k = \alpha\mathbf{x}_i + \beta\mathbf{x}_j$ for some $\alpha, \beta > 0$, $\alpha + \beta = 1$ and some $i, j, k \in \mathbb{N}$ with $i < j < k$ and $\mathbf{x}_k = \frac{\boldsymbol{\tau}_{y_k}\cdots\boldsymbol{\tau}_{y_1}\mathbf{x}_0}{\boldsymbol{\sigma}^T\boldsymbol{\tau}_{y_k}\cdots\boldsymbol{\tau}_{y_1}\mathbf{x}_0}$ for some sequence of observations $Y = y_1, \ldots, y_k$ (and similarly for $\mathbf{x}_j$ and $\mathbf{x}_i$ which truncate the observations at y_j and y_i respectively).
Suppose there exists an equivalent NOOM (represented in its vectorized form) $\mathcal{M}' = (\mathbb{R}^n, \vec{\mathbb{I}}, \{\boldsymbol{\phi}_y\}, \boldsymbol{\psi}_0)$. Let $\mathcal{A}' =$

$(\mathbb{R}^p, \boldsymbol{\sigma}', \{\boldsymbol{\tau}'_y\}_{y\in\mathcal{O}}, \mathbf{x}'_0)$ be a minimal PSR computing the same distribution as $\mathcal{M}'$.
Then, by Lemma 1, we have some similarity transform $\mathbf{S}$ such that $\boldsymbol{\sigma}'^T = \boldsymbol{\sigma}^T\mathbf{S}$, $\boldsymbol{\tau}'_y = \mathbf{S}^{-1}\boldsymbol{\tau}_y\mathbf{S}$, and $\mathbf{x}'_0 = \mathbf{S}^{-1}\mathbf{x}_0$. 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 $\vec{\mathbb{1}}^T\Phi\Pi^+\mathbf{S}^{-1} = \boldsymbol{\sigma}^T, \boldsymbol{\tau}_y = \mathbf{S}\Pi\Phi^+\boldsymbol{\phi}_y\Phi\Pi^+\mathbf{S}^{-1}$, and $\mathbf{x}_0 = \mathbf{S}\Pi\Phi^+\boldsymbol{\psi}_0$.
Now, by the NOOM evolution rules, the NOOM state at timestep k for the sequence Y is $\boldsymbol{\psi}_k = \frac{\boldsymbol{\phi}_{y_k}\cdots\boldsymbol{\phi}_{y_1}\boldsymbol{\psi}_0}{\vec{\mathbb{1}}^T\boldsymbol{\phi}_{y_k}\cdots\boldsymbol{\phi}_{y_1}\boldsymbol{\psi}_0}$ and the probability of any given observation at that time-step is $P(y|\boldsymbol{\psi}_k) = \vec{\mathbb{1}}^T\boldsymbol{\phi}_y\boldsymbol{\psi}_k$.
Further, note that the NOOM state must be a vectorized rank-1 matrix whose eigenvector has unit ℓ_2 norm, i.e., $\boldsymbol{\psi}_k = \mathtt{vec}(\boldsymbol{\psi}_k\boldsymbol{\psi}_k^T)$ with $\|\boldsymbol{\psi}_k\|_2 = 1$.
But we can also write NOOM state in terms of its equivalent PSR-HMM as

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

If $\boldsymbol{\psi}_i$ and $\boldsymbol{\psi}_j$ are valid NOOM states, we have that $\boldsymbol{\psi}_k = \alpha\boldsymbol{\psi}_i + \beta\boldsymbol{\psi}_j = \alpha\mathtt{vec}(\boldsymbol{\psi}_i\boldsymbol{\psi}_i^T) + \beta\mathtt{vec}(\boldsymbol{\psi}_j\boldsymbol{\psi}_j^T) = \mathtt{vec}(\alpha\vec{\psi}_i\boldsymbol{\psi}_i^T + \beta\boldsymbol{\psi}_j\boldsymbol{\psi}_j^T)$. However, $\alpha\boldsymbol{\psi}_i + \beta\boldsymbol{\psi}_j$ is not the vectorization of a rank-1 matrix in general. In particular, $\boldsymbol{\psi}_k$ has unit Kraus-rank only if $\boldsymbol{\psi}_i$ and $\boldsymbol{\psi}_j$ are linearly dependent, which is true only if $\mathbf{x}_i$ and $\mathbf{x}_j$ are linearly dependent. Thus, whenever $\mathbf{x}_i$ and $\mathbf{x}_j$ are linearly independent, $\boldsymbol{\psi}_k$ cannot have unit Kraus-rank. But as normalized HMM states, $\mathbf{x}_i$ and $\mathbf{x}_j$ 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 Kraus-rank, and hence there cannot be an equivalent NOOM. Here, we provide an example of an

HMM that can have a state be a linear (here, convex) combination of two prior linearly independent states

$$\mathbf{x}_0 = \begin{bmatrix}1\\0\end{bmatrix} \quad \boldsymbol{\tau}_1 = \begin{bmatrix}0.25 & 0.5\\0.75 & 0\end{bmatrix} \quad \boldsymbol{\tau_2} = \begin{bmatrix}0 & 0\\0 & 0.5\end{bmatrix} \tag{5.4}$$

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

$$\mathbf{x}_0 = \begin{bmatrix}1\\0\end{bmatrix} \quad \mathbf{x}_1 = \begin{bmatrix}0.25\\0.75\end{bmatrix} \quad \mathbf{x}_2 = \begin{bmatrix}0.7\\0.3\end{bmatrix} = 0.6\mathbf{x}_0 + 0.4\mathbf{x}_1 \tag{5.5}$$

Since $\mathbf{x}_0$ and $\mathbf{x}_1$ 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.

As illustrated by the above example, we could devise a model class more expressive than NOOMs if we simply dispensed with the rank-1 constraint on NOOM states. We now demonstrate how HQMMs can be developed as exactly such models.

5.4 *Liouville HQMMS*

Previous work by Srinivasan et al. (2018e) derived HQMMs by generalizing HMMs using system-environment interactions (illustrated using a quantum circuit), and showed that every n-dimensional HMM can be modeled by an HQMM with no more than an n^2-dimensional hidden states. Here, we take a different approach; we will show how HQMMs can be defined through a series of natural generalizations of NOOMs in such a way that they also end up containing finite-dimensional HMMs. We do so by allowing parameters to be complex and expanding the concepts of NOOM states and operators using the representation in Equation 5.3.

Generalizing NOOM States We know from Equation 5.3 that the initial state $\boldsymbol{\omega}_0$ can viewed as a vectorized rank-1 Hermitian matrix $\hat{\boldsymbol{\rho}}_0$, i.e., $\boldsymbol{\omega}_0 = \mathtt{vec}\left(\mathbf{v}_0\mathbf{v}_0^\dagger\right) = \mathtt{vec}(\hat{\boldsymbol{\rho}}_0)$. A natural generalization would be to let the initial state be a vectorized matrix of arbitrary rank, i.e., $\hat{\boldsymbol{\rho}}_0 = \sum_i p_i \mathbf{v}_i \mathbf{v}_i^\dagger$ instead. The normalization condition on the initial state can then be restated[2] as $1 = \boldsymbol{\sigma}^\dagger \vec{\boldsymbol{\rho}}_0 = \vec{\mathbb{I}}_{n^2}^T \vec{\boldsymbol{\rho}}_0 = \mathrm{tr}\left(\hat{\boldsymbol{\rho}}_0\right) = \sum_i p_i$.

As a linear combination of outer products of vectors with themselves, $\hat{\boldsymbol{\rho}}_0$ must be Hermitian. We additionally assume that the constituent eigenvectors live in a Hilbert space $\mathcal{H}$, so that $\hat{\boldsymbol{\rho}}_0$ lives in a Liouville space, i.e., the outer product of two Hilbert spaces. Further, in

[2] We use $\vec{\boldsymbol{\rho}}$ to represent the vectorization of $\hat{\boldsymbol{\rho}}$, i.e., $\vec{\boldsymbol{\rho}} = \mathtt{vec}(\hat{\boldsymbol{\rho}})$

the NOOM, $\mathbf{v}_0\mathbf{v}_0^\dagger$ had a single eigenvalue of 1. 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 the evaluation $\boldsymbol{\sigma}$, and hence a non-constructive model. Thus, we impose a positive semi-definiteness constraint on the initial state to guarantee that $p_i \in \mathbb{R}_{\geq 0}$ so that $\mathrm{tr}(\hat{\boldsymbol{\rho}}_0)$ is real and non-negative. Essentially, we are now considering a model whose initial states $\vec{\boldsymbol{\rho}}_0$ are vectorized arbitrary-rank Hermitian PSD matrices, which constitute a pointed convex cone. Such matrices are called *density matrices* in quantum mechanics (Nielsen and Chuang, 2010), and the imposition of the PSD constraint on the states is what allows these models to avoid the NPP.

Generalizing NOOM operators Having defined a convex cone of valid states, we now derive operators that ensure that the state always evolves inside the cone. We refer to such operators acting on our states in Liouville space as *Liouville superoperators* $\{\mathbf{L}_y\}_{y\in\mathcal{O}}$. Condition (ii) in Definition 19 ensured that probabilities of observations computed by the NOOM were normalized, and the equivalent condition in the OOM representation in Equation 5.3 is that $\boldsymbol{\sigma}^\dagger \left(\sum_{y\in\mathcal{O}} \boldsymbol{\tau}_y\right) = \boldsymbol{\sigma}^\dagger$. We impose a similar constraint (trace preservation or TP) on the superoperators to ensure we get a normalized distribution over observations. In addition to this, we further need to ensure that the probabilities assigned to observations are real and non-negative, i.e., the operators must always preserve the Hermitian PSD condition of the state.

Finding a constructive way to impose the above restrictions on Liouville superoperators is challenging, and it is easier to do so on the 'reshuffled' version of it called its *Choi matrix* (Wood et al., 2015a). The reshuffle operation involves reshaping the n^2-dimensional columns of the Liouville superoperator into $n \times n$ matrices. Going across the columns of $\mathbf{L}_y$ from left to right, we fill up the blocks of the Choi matrix column-first with these reshaped matrices (Życzkowski and Bengtsson, 2004). In the context of Hermiticity preserving (HP) maps, there is no elegant way to also impose a simple PSD-preserving 'positivity' constraint (Choi, 1975; Pillis, 1967). Therefore, we must impose a slightly more restrictive complete positivity (CP) constraint which guarantees that the map $\mathbf{L}_y \otimes \mathbb{I}$ is PSD-preserving for identity matrices of any dimension. In fact, Choi (1975) suggest that a CP map is the natural constructive generalization of 'positivity' for a linear HP map.

We define L-HQMMs as a generalization of NOOMs with these constraints:

Definition 20 (L-HQMMs (Adhikary et al., 2020))**.** *An n^2-dimensional Liouville-Hidden Quantum Markov Model with a set of discrete observations $\mathcal{O}$ is a tuple $(\mathbb{C}^{n^2}, (\mathbf{L}_y)_{y\in\mathcal{O}}, \vec{\boldsymbol{\rho}}_0, \vec{\mathbb{I}})$ where the initial state $\vec{\boldsymbol{\rho}}_0 \in \mathbb{C}^{n^2}$ 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:*

(i) $\vec{\boldsymbol{\rho}}_0$ *is a vectorized Hermitian PSD matrix of arbitrary rank,*

(ii) Normalized initial state: $\vec{\mathbb{I}}^T\vec{\boldsymbol{\rho}}_0 = 1$,

(iii) CP: $\mathbf{C}_y \geq 0$ *(Choi matrix is PSD).*

(iv) TP: $\vec{\mathbb{I}}^T \left(\sum_{y\in\mathcal{O}} \mathbf{L}_y\right) = \vec{\mathbb{I}}^T$,

(v) HP: $\mathbf{C}_y = \mathbf{C}_y^\dagger$,

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

$$\vec{\boldsymbol{\rho}}_t = \frac{\mathbf{L}_y\vec{\boldsymbol{\rho}}_{t-1}}{\vec{\mathbb{I}}^T\mathbf{L}_y\vec{\boldsymbol{\rho}}_{t-1}} \qquad P(\bar{y}) = \vec{\mathbb{I}}^T\mathbf{L}_{y_t} \ \ldots \ \mathbf{L}_{y_1}\vec{\boldsymbol{\rho}} \tag{5.6}$$

L-HQMMs are equivalent to K-HQMMs Prior work on HQMMs have represented these models in the so-called operator-sum representation (Srinivasan et al., 2018e; Monras et al., 2010). While operations on vectorized matrices are fairly common in quantum information (and was implicitly used for HQMMs in Srinivasan et al. (2018b)), L-HQMMs were a novel formulation of HQMMs introduced by Adhikary et al. (2020). We can derive the operator-sum representation of K-HQMMs from L-HQMMs, showing that the two are equivalent.

From Definition 20, we know that any model equivalent to L-HQMMs must have CP, TP, and HP operators. From Choi's theorem (Choi, 1975), we know that any map which can be expressed in the operator-sum representation $\mathcal{K}(\hat{\boldsymbol{\rho}}) = \sum_w \mathbf{K}_w \hat{\boldsymbol{\rho}} \mathbf{K}_w^\dagger$ is guaranteed to be CP, and will preserve the PSD nature of any input matrix. In the context of CP maps, the operator matrices $\mathbf{K}_w$ are commonly called Kraus operators (Kraus, 1971). The quadratic application of operators preserves the Hermiticity of $\hat{\boldsymbol{\rho}}$. Thus, the operator-sum representation is particularly appealing because it guarantees the CP and HP constraints by construction. Note that this representation of CP maps is merely the inverse vectorization of the Liouville form

$$\mathsf{vec}\left(\sum_w \mathbf{K}_w \hat{\boldsymbol{\rho}} \mathbf{K}_w^\dagger\right) = \sum_w (\mathbf{K}_w^* \otimes \mathbf{K}_w)\vec{\boldsymbol{\rho}} = \mathbf{L}\vec{\boldsymbol{\rho}}$$

Thus, the action of a Liouville superoperator $\mathbf{L}_y$ corresponding to the observable y on $\vec{\boldsymbol{\rho}}$ can can be equivalently represented by a set

of Kraus operators $\{\mathbf{K}_{y,w_y}\}$ acting on the density matrix $\hat{\boldsymbol{\rho}}$, where the cardinality of this set $|w_y|$ is determined by the Schmidt-rank (or Kraus-rank, as we soon explain) of $\mathbf{L}_y$. The Schmidt-rank is analogous to the rank revealed by an SVD, but for a decomposition into a Kronecker product of two vector spaces.

Uniqueness of L-HQMMs While L-HQMMs (Definition 20) and and K-HQMMs (Definition 11) are equivalent representations of HQMMs, the former has the added benefit of providing a unique representation of the underlying CP map. Namely, a CP map can be equivalently defined using Kraus operator sets, which may not even have the same number of operators. This makes it difficult to directly compare two K-HQMM models, perhaps to check for equivalency. In contrast, the Liouville superoperator of a CP-map is unique, and can be canonically factorized as follows (Wood et al., 2015a; Miszczak, 2011):

$$\mathbf{L} \;=\; \sum_{w} \mathbf{K}_w^* \otimes \mathbf{K}_w \;=\; \sum_{i=1}^{r} \gamma_i (\mathbf{K}_i^* \otimes \mathbf{K}_i) \tag{5.7}$$

where $\{\mathbf{K}_w\}$ is a set of arbitrary Kraus operators, $\{\sqrt{\gamma_i}\,\mathbf{K}_i\}$ the set of *canonical* Kraus operators defining the CP map, and r the 'Kraus-rank' of the CP map. What we require here is essentially a Kronecker Product SVD (KPSVD) (Golub and Loan, 1996) of $\mathbf{L}$ where the Kronecker product factors are restricted to have a special symmetric form. It is a well known result that these factors can be computed directly from an SVD of the Choi matrix (the 'reshuffled' Liouville matrix); the i-th singular value and vector pair correspond to γ_i and $\mathrm{vec}(\mathbf{K}_i)$ (Wood et al., 2015a; Miszczak, 2011). We illustrate this process in Figure 5.1.

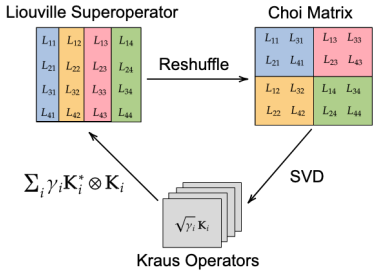


Figure 5.1: **Three equivalent formulations of a CP map**: The unique canonical operator sum representation of a CP map can be obtained by performing an SVD of its Choi matrix, which is obtained by reshuffling its Liouville superoperator.

The Kraus-rank of a CP map is equal to the rank of the Choi matrix, and is equal to the minimum number of Kraus operators required to express the operation. Since the Liouville superoperator (or the Choi matrix) uniquely defines a CP map, we can use this representations to compare two L-HQMMs. This also provides a way to compare two K-HQMMs by first converting them to their corresponding L-HQMMs.

HQMMs are OOMs The exact relationship between HQMMs and OOMs was previously unknown, but the Liouville formulation of HQMMs allowed us to state an important result in Adhikary et al. (2020):

Theorem 7. *HQMM $\subset$ OOM, and the set of valid initial states for HQMMs is a spectraplex.*

Proof. Setting $\boldsymbol{\sigma} = \vec{\mathbb{I}}$, L-HQMMs satisfy condition (i) of General OOMs laid out in Definition 17 by construction. Condition (ii) of Definition 17 is satisfied by the TP constraint on L-HQMMs. Next, the HP and CP constraints on L-HQMMs guarantee that $\mathbf{L}_{\bar{y}}\vec{\boldsymbol{\rho}}$ always yields a vectorized Hermitian PSD matrix. The trace of this matrix is always real and non-negative, i.e., $\vec{\mathbb{I}}^T\mathbf{L}_{\bar{y}}\vec{\boldsymbol{\rho}} \geq 0$. We also have $\vec{\mathbb{I}}^T\mathbf{L}_y\vec{\boldsymbol{\rho}}_0 \leq \vec{\mathbb{I}}^T\left(\sum_{y\in\mathcal{O}}\mathbf{L}_y\right)\vec{\boldsymbol{\rho}}_0 = \vec{\mathbb{I}}^T\vec{\boldsymbol{\rho}}_0 = 1$, satisfying condition (iii) of Definition 17.
The valid initial states of L-HQMMs are Hermitian PSD matrices with unit trace. 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. Using the same similarity transform that we used for NOOMs $\mathbf{S} = \mathbb{I}_{n^2} + \frac{1}{n^2}\vec{\mathbf{1}}_{n^2}(\boldsymbol{\sigma}^T - \vec{\mathbf{1}}_{n^2}^T)$, we can transform any n^2-dimensional L-HQMM into an equivalent standard OOM.
The above proof establishes that HQMM $\subseteq$ OOM. Recently, Fanizza et al. (2024) have shown that there exist linear stochastic processes that do not admit quantum realization, which allows us to establish the strict relationship HQMM $\subset$ OOM. □

State and Operator Space Geometries of NOOMs We have shown that n-dimensional NOOMs form a subset of n-dimensional HQMMs through generalization. Prior work by Srinivasan et al. (2018e) used 'HQMMs' and 'NOOMs' somewhat ambiguously, differentiating them primarily by the field over which they are defined ($\mathbb{R}$ or $\mathbb{C}$). In Adhikary et al. (2020), we use the original formulation of NOOMs (Zhao and Jaeger, 2010b) to draw a clearer distinction, whereby NOOMs are simply HQMMs with rank-1 vectorized initial states and Kraus-rank 1 operators. Particularly, for a fixed latent dimension n^2 of the vectorized density matrix, an HQMM allows for a greater diversity of both states and dynamics.

First, note that the valid states of HQMMs are Hermitian PSD matrices with unit trace, also known as mixed density matrices in quantum mechanics (Nielsen and Chuang, 2010). By contrast, the valid states for NOOMs correspond to the set of pure density operators (with rank 1). Since these operators encode the probability distribution of the latent state, we see that HQMM states can represent mixture distributions of NOOM states. Geometrically, the set of rank-1 density matrices are extremal points of the spectraplex defined by arbitrary rank density matrices, i.e., NOOM states define the shell of the spectralex of HQMM states. This gives us some geometric intuition for why HQMMs have a richer state space than NOOMs.

Second, HQMMs can have an arbitrary number of Kraus operators

per observable while NOOMs are restricted to one to preserve rank-1 states. This indicates that the evolution associated with individual observations in an n-dimensional NOOM is restricted to dynamics corresponding to rank-1 Choi matrices. Thus, an n-dimensional HQMMs with arbitrary Kraus rank can encode richer dynamics than an n-dimensional NOOM.

NOOM $\subset$ HQMM Since we defined L-HQMMs as generalizations of NOOMs, we would expect that HQMMs form a superset of NOOMs. Indeed, as we showed in Adhikary et al. (2021a), this follows from the results we have already established:

Theorem 8 (NOOM $\subset$ HQMM)**.** *(Adhikary et al., 2021a) Finite dimensional norm-observable operator models are a strict subset of finite dimensional hidden quantum Markov models.*

Proof. We know from Theorem 5 that there exist HMMs that cannot be modeled by finite-dimensional NOOMs. However, since all HMMs can be modeled by finite dimensional HQMMs, the same HMMs serve as instances of HQMMs that cannot be modeled by NOOMs. This, combined with the fact that NOOM $\subseteq$ HQMM (Adhikary et al., 2020), give us that NOOM $\subset$ HQMM. □

5.5 Conclusion

We analyzed how the increased expressiveness of HQMMs stems from relaxing non-negativity constraints on states and operators—an idea motivated by quantum operations but not limited to them. Specifically, we demonstrated that HQMMs are OOMs, which do not suffer from the negative probability problem. Additionally, we also established two important relationships about an alternative model class from the classical stochastic process literature. Specifically, we showed that NOOMs form a relatively restrictive model class and do not contain HMMs as a subclass – there exist finite dimensional HMMs that cannot be modeled exactly by finite dimensional NOOMs. However, as we demonstrated, HQMMs *do* subsume both of these classes.

Equipped with the model relationships demonstrated in this chapter, we are able to concisely visualize the hierarchy of expressiveness between the models we have considered in Figure 5.5. In the gray box, we list the novel relationships we have established.

In light of the expressiveness hierarchy in Figure 5.5, we see that HQMMs appear as an attractive model class for linear stochastic

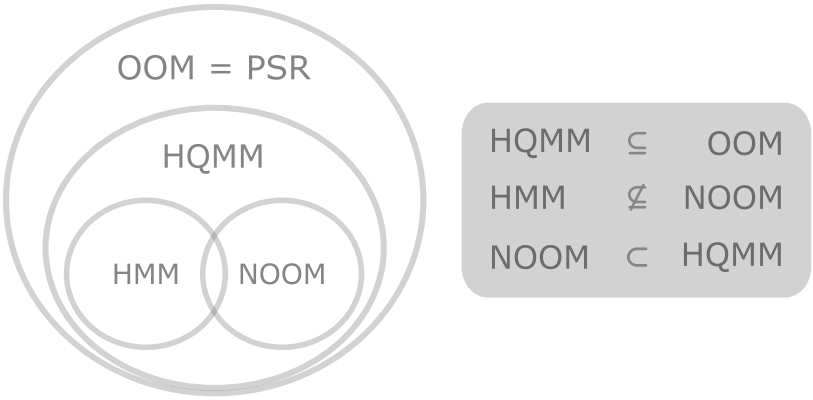


Figure 5.2: **Hierarchy of expressiveness of linear sequential systems** Subset relationships denote relative expressiveness reltionships from Definition 12. The gray box indicates new relationships established in Adhikary et al. (2020), and summarized in this chapter.

processes – one that escapes negative probabilities while retaining more expressiveness than other alternatives. While OOMs/PSRs are guaranteed to be expressive enough to model any linear stochastic process, their application has been limited to some extent due to their tendency to yield negative probabilities. Some approaches to mitigate this issue have included heuristics like flipping the sign of negative probabilities or projecting a learned PSR to the nearest valid HMM (Balle et al., 2014a; Cohen et al., 2013). However, such heuristics break convergence guarantees of learning algorithms, and projections nullify the benefit of picking an expressive model class in the first place. On the other hand, we are able to learn HQMMs reliably without needing such heuristics, for instance with the learning algorithm introduced in Chapter 4.

6
HQMMs with Control

Abstract *We expand our analysis of the expressiveness of hidden quantum Markov models (HQMMs) to controlled stochastic processes, and introduce Input-Output (IO) HQMMs that include a notion of action or control – establishing these models as special cases of the larger class of IO-OOMs. Just as HQMMs were proposed as quantum-inspired generalization of HMMs, quantum-inspired variants of controlled HMMs or partially observable Markov decision processes (POMDPs) have also been proposed in the literature including quantum obsersavable Markov decision processes (QOMDPs) (Barry et al., 2014; Ying and Ying, 2014) and quantum Markov decision processes (QuaMDPs) (Cidre, 2016). We derive a series of expressiveness relationships to demonstrate how these quantum-inspired models also neatly fit into the spectrum of IO-OOMs, and specifically are subsumed by IO-HQMMs. Finally, we use these established subset relationships to prove that perfect goal state reachability is undecidable for IO-OOMs, by generalizing the same result proven in the quantum-MDP literature.*

In our analysis of models for linear sequential systems in Chapter 3 and Chapter 5, we have restricted ourselves to *uncontrolled* systems. In this setting, we assume that we only have passive access to the stochastic process being modeled via sequences of observations. In the *controlled* setting, we add an additional notion of control via actions that can perturb the process. This addition expands the scope of the problem setting from probabilistic sequential prediction to include probabilistic sequential decision making – identifying a sequence of actions that induces some desired behavior in a stochastic process. Most of the models we have considered thus far have a controlled counterpart – including HMMs, OOMs, and PSRs. In this chapter, we propose a controlled counterpart to HQMMs, namely Input-Output HQMMs, following similar formulations proposed for OOMs.

Similar to how HQMMs were proposed as generalizations of HMMs, there have been similar proposals of quantum-inspired generalizations of the controlled counterpart of HMMs known as partially observable Markov decision processes (POMDPs). These

include quantum observable Markov decision processes (QOMDPs) Barry et al. (2014); Ying and Ying (2014) and quantum Markov decision processes (QuaMDPs) proposed by Cidre (2016). Additionally, Barry et al. (2014) show that QOMDPs are not simply a quantum-translation of POMDPs, but have interesting new properties. Namely, Barry et al. (2014) show that while perfect goal state reachability is undecidable for QOMDPS, even though they are decidable for POMDPs. By incorporating these models into an expressiveness hierarchy as in Chapter 5, we are able to establish the same property for classical liner sequential systems as well.

Outline We begin our analysis in Section 6.1 with background on POMDPs, which can be viewed as controlled counterpart of HMMs. In Section 6.2, we summarize input-output OOMs, controlled variants of general OOMs discussed in Chapter 5. Following this formulation, we introduce input-output HQMMs in Section 6.3, and demonstrate how they subsume QOMDPs and QuaMDPs described in Section 6.4. Finally, in Section 6.5, we demonstrate the undecidabiliy of perfect planning in IO-OOMs using the expressiveness relationships we have established.

Notation We use the same mathematical notation used in the earlier Chapters as described in Section 3.1.

6.1 *Partially Observable Markov Decision Processes*

Markov decision processes form the central construct in most applications of probabilistic sequential decision making. An MDP can be defined as a tuple $(\mathcal{X}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{X}$ and $\mathcal{A}$ are discrete[1] state and action spaces, $\mathcal{R} : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is a discount factor. The distribution $P^a_{\mathbf{x}} = P(\mathbf{x}'|\mathbf{x}, a)$ encodes the probability of transitioning from state $\mathbf{x}$ to $\mathbf{x}'$ under action a. The RL objective is to train a policy mapping states to a distribution over actions, that maximizes the expected discounted return $\mathbb{E}_\pi\left[\sum_{t=0}^{H} \gamma^t r^a_{\mathbf{x}_t}\right]$, where $r^a_{\mathbf{x}} = \mathcal{R}(\mathbf{x}, a)$, and H is the (potentially infinite) time horizon. The expectation $\mathbb{E}_\pi$ is taken over $P^\pi_{\mathbf{x}}$, the transition distribution under the policy π. While the reward function and its induced policy are generally of primary interest in learning control strategies, we will choose to ignore[2] these for our analyses, and consider MDPs as tuples $(\mathcal{X}, \mathcal{A}, \mathcal{P})$. Moreover, given a discrete state and action spaces, the probabilistic transition model $\mathcal{P}$ can be encoded into a set of transition matrices $\{\mathbf{A}^a \in \mathbb{R}^{n \times n}_{\geq 0}\}$ where $\mathbf{A}^a$ holds the transition probabilities $P^a_{\mathbf{x}}$ for action a.

In formulating MDPs, we have inherently assumed that states

[1] In general, MDPs can be defined over continuous state and action spaces. We restrict ourselves to the discrete setting as we have done in the controlled case in Chapter 5

[2] We ignore the reward function to draw a clearer comparison with uncontrolled processes. Our analysis can be equivalently applied with reward functions.

$\mathbf{x} \in \mathcal{X}$ can be directly observed. However, this is generally not the case in most applications – we often only have access to noisy observations derived from these states[3]. Thus, besides the transition model $\{\mathbf{A}^a\}$, we also need to introduce a probabilistic emission model $\{\mathbf{C}^a\}$ defined over an observation space $\mathcal{O}$ that encodes probabilities $P(y|\mathbf{x}, a)$ of observing $y \in \mathcal{O}$ conditioned on taking action a at state $\mathbf{x}$. With this addition, we arrive at partially observable Markov decision processes, which we define as follows:

[3] In practice, it is common to essentially ignore this limitation and treat observations as states, or stack observations over multiple prior time steps as the current state. The idea behind the latter approach being that the information contained in the state can often be extracted from a history of observations.

Definition 21 (Partially Observable Markov Decision Processes)**.** *An n-dimensional partially observable Markov decision process for sets of discrete actions $\mathcal{A}$ and observations $\mathcal{O}$ is a tuple $(\mathbb{R}^n, \{\mathbf{A}^a\}_{a\in\mathcal{A}}, \{\mathbf{C}^a\}_{a\in\mathcal{A}}, \mathbf{x}_0)$, where the matrices $\mathbf{A}^a \in \mathbb{R}_{\geq 0}^{n\times n}$ and $\mathbf{C}^a \in \mathbb{R}_{\geq 0}^{|\mathcal{O}|\times n}$ are non-negative column-stochastic transition and emission matrices corresponding to the action a such that $\vec{\mathbb{1}}^T\mathbf{A}^a = \vec{\mathbb{1}}^T\mathbf{C}^a = \vec{\mathbb{1}}^T$. The initial state $\mathbf{x}_0 \in \mathbb{R}_{\geq 0}^n$ is also non-negative and satisfies $|\mathbf{x}_0|_1 = 1$.*

As with HMMs, POMDPs can also be defined via observable operators

Definition 22 (POMDPs as OOMs)**.** *Given an n-dimensional partially observable Markov decision process $(\mathbb{R}^n, \{\mathbf{A}^a\}_{a\in\mathcal{A}}, \{\mathbf{C}^a\}_{a\in\mathcal{A}}, \mathbf{x}_0)$, its observable operator $\mathbf{T}_y^a$ corresponding to taking action a and observing y is defined as $\mathbf{T}_y^a = \texttt{diag}\big(\mathbf{C}_{(y,:)}^a\big)\mathbf{A}^a$, where the matrix $\mathbf{C}_{(y,:)}^a$ is a diagonal matrix formed from the y-th row of $\mathbf{C}^a$. The state update is done as $\mathbf{x}_t = \frac{\mathbf{T}_y^a\,\mathbf{x}_{t-1}}{\vec{\mathbb{1}}^T\mathbf{T}_y^a\,\mathbf{x}_{t-1}}$. The probability of observing y_t after taking action a_t at each timestep is computed as $P(a_1y_1, \ldots, a_N, yN) = \vec{\mathbb{1}}^T\mathbf{T}_{y_t}^{a_t}\ldots\mathbf{T}_{y_1}^{a_1}\mathbf{x}$.*

6.2 Input-Output OOMs (IO-OOMs)

By augmenting the *uncontrolled* OOMs with a notion of control, Jaeger (1998) developed Input-Output OOMs (IO-OOMs) by essentially defining a separate OOM for each action. As with standard OOMs in Definition 16, we define these models as standard IO-OOMs as follows:

Definition 23 (Standard IO-OOMs (Jaeger, 1998))**.** *An n-dimensional standard input-output observable operator model with sets of discrete actions $\mathcal{A}$ and observations $\mathcal{O}$ is a tuple $(\mathbb{R}^n, \{\mathbf{T}_y^a\}_{y\in\mathcal{O},a\in\mathcal{A}}, \mathbf{x}_0)$ where initial state $\mathbf{x}_0 \in \mathbb{R}^n$ and observable operators $\{\mathbf{T}_y\}_{y\in\mathcal{O}} \in \mathbb{R}^{n\times n}$ satisfy the following constraints:*

(i) Normalized initial state: $\vec{\mathbb{1}}^T\mathbf{x}_0 = 1$,

(ii) Normalized marginal over observations for every action: $\forall a \in \mathcal{A}$ $\vec{\mathbb{1}}^T\sum_{y\in\mathcal{O}}\mathbf{T}_y^a = \vec{\mathbb{1}}^T$,

(iii) Non-negative probabilities: $\vec{\mathbb{1}}^T \mathbf{T}_{y_t} \dots \mathbf{T}_{y_1} \mathbf{x}_0 \geq 0$ *for all observation-action sequences* $\{(a_1, y_1), \dots, (a_t, y_t)\}$

Similar to the development of general OOMs from standard OOMs in Section 5.2), Thon and Jaeger (2015) generalize standard IO-OOMs as generic controlled linear sequential systems. We define these models as general IO-OOMs as follows:

Definition 24 (General Input-Output OOMs (Thon and Jaeger, 2015))**.** *An n-dimensional general input-output observable operator model for sets of discrete actions* $\mathcal{A}$ *and observations* $\mathcal{O}$ *is a tuple* $(\mathbb{C}^n, \{\boldsymbol{\tau}_y^a\}_{y \in \mathcal{O}, a \in \mathcal{A}}, \mathbf{x}_0, \boldsymbol{\sigma})$ *where initial state* $\mathbf{x}_0 \in \mathbb{C}^n$*, observable operators* $\{\boldsymbol{\tau}_y^a\} \in \mathbb{C}^{n \times n}$*, and a linear evaluation functional* $\boldsymbol{\sigma} \in \mathbb{C}^n$ *satisfy the following constraints:*

(i) Normalized Initial State: $\boldsymbol{\sigma}^\dagger \mathbf{x}_0 = 1$*,*

(ii) Normalized marginal over observations and actions: $\boldsymbol{\sigma}^\dagger \boldsymbol{\tau}_{y_t}^{a_t} \dots \boldsymbol{\tau}_{y_1}^{a_1} x_0 = \sum_{y \in \mathcal{O}} \boldsymbol{\sigma}^\dagger \boldsymbol{\tau}_y^a \boldsymbol{\tau}_{y_t}^{y_t} \dots \boldsymbol{\tau}_{y_1}^{a_1} \mathbf{x}_0$ *for all observation-action sequences* $\{(a_1, y_1), \dots, (a_t, y_t)\}$*.*

(iii) Non-negative probabilities: $\boldsymbol{\sigma}^\dagger \boldsymbol{\tau}_{y_t}^{a_t} \dots \boldsymbol{\tau}_{y_1}^{a_1} \mathbf{x}_0 \in [0, 1]$ *for all observation-action sequences* $\{(a_1, y_1), \dots, (a_t, y_t)\}$*.*

6.3 Input-Output HQMMs (IO-HQMMs)

The class of general IO-OOMs developed by Thon and Jaeger (2015) are equivalent to controlled PSRs, and provide a unified framework for controlled linear sequential systems. Moreover, just as OOMs are strictly more expressive than HMMs (HMM $\subset$ OOM), IO-OOMs are strictly more expressive than POMDPs (POMDPs $\subset$ IO-OOM). To incorporate HQMMs into this hierarchy, we now define input-output HQMMs as follows

Definition 25 (Input-Output HQMM)**.** *An* n^2*-dimensional input-output hidden quantum Markov model for sets of discrete actions* $\mathcal{A}$ *and observations* $\mathcal{O}$ *is a controlled stochastic process given by the tuple* $(\mathbb{C}^{n^2}, \vec{\mathbb{I}}, \{\mathbf{L}_y^a\}_{y \in \mathcal{O}, a \in \mathcal{A}}, \vec{\boldsymbol{\rho}}_0)$*. The set of operators* $\{\mathbf{L}_y^a\}_{y \in \mathcal{O}}$ *for every action a form a set of CP-TP Liouville operators. The initial state* $\vec{\boldsymbol{\rho}}_0$ *is a vectorized unit-trace Hermitian PSD matrix of arbitrary rank.*

The state update and probability computations for IO-HQMMs are done exactly the same as with HQMMs, but now with operators indexed by both observations and actions at each time step:

$$P(a_1 y_1 \dots a_t y_t) = \vec{\mathbb{I}}^T \mathbf{L}_{y_t}^{a_t} \dots \mathbf{L}_{y_1}^{a_1} \vec{\boldsymbol{\rho}}_0 \quad \text{and} \quad \vec{\boldsymbol{\rho}}\,' = \frac{\mathbf{L}_{y_t}^{a_t} \dots \mathbf{L}_{y_1}^{a_1} \vec{\boldsymbol{\rho}}_0}{\vec{\mathbb{I}}^T \mathbf{L}_{y_t}^{a_t} \dots \mathbf{L}_{y_1}^{a_1} \vec{\boldsymbol{\rho}}_0}$$

Besides IO-HQMMs, there have been a few other quantum-inspired probabilistic models for controlled linear stochastic processes. We

now discuss these models, and show that they are subsumed by IO-HQMMs in terms of expressiveness.

6.4 *Quantum Observable Markov Decision Processes*

Barry et al. (2014) developed quantum observable Markov decision processes (QOMDPs) as a generalization of classical POMDPs, by swapping the belief state vector $\mathbf{x}$ with the density matrix of a quantum state $\hat{\boldsymbol{\rho}}$. Ignoring reward functions, QOMDPs can be defined as follows

Definition 26 (QOMDPs (Barry et al., 2014))**.** *An n-dimensional quantum observable Markov decision process for sets of discrete actions $\mathcal{A}$ and observations $\mathcal{O}$ is a tuple $(\mathbb{C}^n, \{\mathbf{K}_y^a\}_{a\in\mathcal{A},y\in\mathcal{O}}, \hat{\boldsymbol{\rho}}_0)$, where the set of operators $\{\mathbf{K}_y^a \in \mathbb{C}^{n\times n}\}_{y\in\mathcal{O}}$ for each action forms a quantum channel with $\sum_{y\in\mathcal{O}} \mathbf{K}_y^{a\dagger}\mathbf{K}_y^a = \mathbb{I}$. The initial state $\hat{\boldsymbol{\rho}}_0$ is a unit-trace Hermitian PSD density matrix of arbitrary rank.*

The probability of an action-observation sequence $a_1y_1\cdots a_Ty_T$ for a QOMDP is then taken to be $\mathrm{Tr}\left(\mathbf{K}_{y_T}^{a_T}\ldots\mathbf{K}_{y_1}^{a_1}\,\hat{\boldsymbol{\rho}}_0\,\mathbf{K}_{y_1}^{a_1\dagger}\ldots\mathbf{K}_{y_T}^{a_T\dagger}\right)$. To draw connections with HQMMs, we represent the vectorized version of the QOMDP update equations as follows:

$$\begin{aligned}\vec{\boldsymbol{\rho}}\,' &= \mathrm{vec}\left(\frac{\mathbf{K}_{o_T}^{a_T}\ldots\mathbf{K}_{o_1}^{a_1}\,\hat{\boldsymbol{\rho}}\,\mathbf{K}_{o_1}^{a_1\dagger}\ldots\mathbf{K}_{o_T}^{a_T\dagger}}{\mathrm{Tr}\left(\mathbf{K}_{o_T}^{a_T}\ldots\mathbf{K}_{o_1}^{a_1}\,\hat{\boldsymbol{\rho}}\,\mathbf{K}_{o_1}^{a_1\dagger}\ldots\mathbf{K}_{o_T}^{a_T\dagger}\right)}\right)\\ &= \frac{\left(\overline{\mathbf{K}}_{y_T}^{a_T}\otimes\mathbf{K}_{y_T}^{a_T}\right)\ldots\left(\overline{\mathbf{K}}_{y_1}^{a_1}\otimes\mathbf{K}_{y_1}^{a_1}\right)\vec{\boldsymbol{\rho}}}{\vec{\mathbb{I}}\left(\overline{\mathbf{K}}_{y_T}^{a_T}\otimes\mathbf{K}_{y_T}^{a_T}\right)\ldots\left(\overline{\mathbf{K}}_{y_1}^{a_1}\otimes\mathbf{K}_{y_1}^{a_1}\right)\vec{\boldsymbol{\rho}}} = \frac{\mathbf{L}_{y_T}^{a_T}\ldots\mathbf{L}_{y_1}^{a_1}\vec{\boldsymbol{\rho}}_0}{\vec{\mathbb{I}}^T\mathbf{L}_{y_T}^{a_T}\ldots\mathbf{L}_{y_1}^{a_1}\vec{\boldsymbol{\rho}}_0}\end{aligned}$$

Notice that this expression is identical to the update equation for IO-HQMMs. The only difference is that the QOMDP operators are restricted to have unit Kraus-rank: $\mathbf{L}_y^a = \overline{\mathbf{K}}_y^a \otimes \mathbf{K}_y^a$. This makes them somewhat like NOOMs; however, unlike NOOMs, QOMDPs allow latent states of arbitrary Kraus-ranks – i.e. both mixed and pure states. We thus arrive at the following theorem characterizing the expressiveness of QOMDPs

Theorem 9. *QOMDPs $\subseteq$ IO-HQMMS $\subset$ IO-OOMs = PSRs*

Proof. We can see that IO-HQMMs $\subset$ IO-OOMs essentially from their definitions – the former prescribes a specific instantiation of the parameters of the latter – and the fact that HQMMs $\subset$ OOMs. To show that QOMDPs $\subseteq$ IO-HQMMs, we note that QOMDPs operators are restricted to have unit Kraus-rank, whereas IO-HQMMs do not have this restriction. □

Alternative Quantum MDPs Quantum Markov decision processes were also studied by Ying and Ying (2014), who arrived at essentially the same model as QOMDPs. Cidre (2016) proposed an alternate formulation, called Quantum MDPs (QuaMDPs), along with an associated point-based value iteration algorithm similar to that used to learn classical POMDPs (Pineau et al., 2003). Although connections to QOMDPs were not considered in the original work, QuaMDPs are a special case of QOMDPs. The filtering process in IO-HQMMs and QOMDPs corresponds to positive operator valued measurements, which are generalizations of the more restrictive projection valued measurements used in QuaMDPs. One can reduce positive operator valued measurements to projection based measurements via the Stinespring dilation theorem (Stinespring, 1955), but this requires the system interacting with an ancillary sub-system; this is not the case in QuaMDPs.

6.5 Undecidability of Perfect Planning in IO-OOMs

In moving from POMDPs to QOMDPs, Barry et al. (2014) find an apparent classical-quantum separation in the problem of perfect goal state reachability. They consider particular instances of controlled processes where certain states are labeled as goals, and are set to be absorbing – the transition probabilities from these states to any other state is zero; we will refer to these as *goal oriented* models. This is a common way to frame planning problems in the absence of reward functions. We define such models as below

Definition 27 (Goal Oriented Decision Process)**.** *A goal oriented decision process consists of an absorbing goal state $\boldsymbol{g}_s$ such that an agent that reaches it will always stay there.*

Given a goal oriented decision processes, the standard problem one faces is to define or learn a policy that takes an agent from some initial state to the goal state in a finite number of steps. Prior to finding such a policy, we need to first answer a more fundamental question: does such a policy even exist? Formally, the perfect goal state reachability problem is defined as follows

Definition 28 (Perfect Goal State Reachability)**.** *Given a goal oriented decision process as defined in Definition 27, the perfect goal state reachability problem is to determine whether there exists a policy that will take an agent to the goal state from an arbitrary initial state.*

Barry et al. (2014) show that this problem is undecidable for QOMDPs, even though it is decidable for POMDPs. The undecidability of perfect goal state reachability in QOMDPs is a consequence

of the undecidability of the quantum measurement occurence problem (Eisert et al., 2012). Intuitively, Barry et al. (2014) point out that the decidability of the same problem for POMDPs boils down to its non-negativity constraints. However, negative parameters are also present in *classical* generalizations of POMDPs, namely IO-OOMs or PSRs, as well as other quantum models including IO-HQMMs. Thus, we would expect this problem should be undecidable for these models as well. Indeed, we can show this to be the case using the subset relationships we have established between these models and QOMDPs.

Theorem 10 (Perfect Goal State Reachability)**.** *Given an initial state and a goal state, it is undecidable whether there exists a policy that will leave a goal-oriented IO-HQMM, IO-OOM or PSR in the goal state in a finite number of steps.*

Proof. QOMDPs are contained within the class of IO-HQMMs, which are, in turn, contained within the class of IO-OOMs or PSRs. If a problem is undecidable for a model class, it must be undecidable in general for any other class that contains it as a subset. Therefore, we immediately see that the undecidability carries over to these models as well. □

6.6 Conclusion

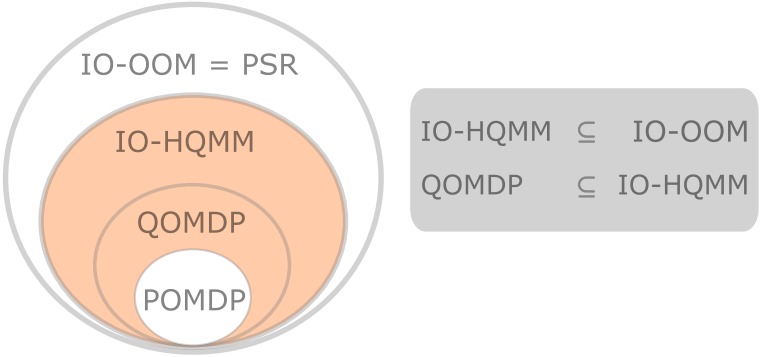


Figure 6.1: **Hierarchy of expressiveness of controlled linear sequential systems** Subset relationships denote relative expressiveness relationships from Definition 12. The gray box indicates new relationships established. The orange (shaded) sections represent potentially non-strict subsets.

We extended the framework of HQMMs to controlled settings by introducing input-output HQMMs (IO-HQMMs), which parallel the development of IO-OOMs in the classical literature. The IO-HQMM model class provided us with a unifying framework that subsumes some existing quantum-inspired models for controlled stochastic processes including quantum observable Markov decision processes (QOMDPs) and Quantum Markov decision processes (QuaMDPs). Specifically, QuaMDPs are special cases of QOMDPs, and, as we

demonstrated, QOMDPs are special cases of IO-HQMMs where operators are restricted to have unit Kraus rank.

Equipped with the model relationships demonstrated in this chapter, we are able to concisely visualize the hierarchy of expressiveness between the models we have considered in Figure 6.1. In the figure, the gray box indicates new relationships established in this chapter, and the orange shaded regions are potentially non-empty, i.e., we have established $\subseteq$ relationships as opposed to $\subset$. Moreover, by establishing QOMDP $\subseteq$ IO-HQMM, we were also able to translate a known property of QOMDPs to the general class of IO-OOMs – perfect goal state reachability is undecidable for IO-OOMs.

7
Uniform Quantum Tensor Networks

Abstract *We expand our analysis of the expressiveness of HQMMs beyond the literature of stochastic processes by incorporating quantum tensor networks (QTNs) and weighted automata (WA). The physics community developed tensor networks known as* matrix product states*, a tensor-train decomposition for probabilistic modeling, motivated by the need to tractably model many-body systems. But similar models have also been studied in the stochastic processes and weighted automata literature, with little work on how these bodies of work relate to each other. We address this gap by showing how stationary or uniform versions of popular quantum tensor network models have equivalent representations in the stochastic processes and weighted automata literature, in the limit of infinitely long sequences. We demonstrate several equivalence results between models used in these three communities: (i) uniform variants of matrix product states, Born machines and locally purified states from the quantum tensor networks literature, (ii) predictive state representations, hidden Markov models, norm-observable operator models and hidden quantum Markov models from the stochastic process literature, and (iii) stochastic weighted automata, probabilistic automata and quadratic automata from the formal languages literature.*

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). But the same problem has also been studied from various other per-

spectives. Notably, the models discussed in Chapter 5 including observable operator models (OOMs) (Jaeger, 2000) or predictive state representations (PSRs) (Singh et al., 2004) from the machine learning literature and stochastic weighted automata (Balle et al., 2014b) from the formal language literature are approaches to tackle essentially the same problem.

While Thon and Jaeger (2015) provide an overview discussing connections between OOMs and stochastic weighted automata, their connection to MPS has not been extensively explored. 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). In Adhikary et al. (2021a), we tackled the *uniform* case, and established similar relationships by examining how various quantum tensor networks relate to aforementioned work in different fields. In this chapter, we derive a collection of results analyzing the relationships in expressiveness between uniform MPS and their various subclasses (Adhikary et al., 2021a).

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 tends to focus more on non-uniform models. We aim to fill this gap by developing expressiveness relationships for uniform variants.

We expect that the connections presented here will also open the door for results and methods in one area to be applied in another. For instance, one of the proof strategies we adopt is to develop expressiveness relationships between subclasses of PSRs, and show how they also carry over to equivalent uniform tensor networks. Such cross fertilization also takes place at the level of algorithms. 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. With the equivalence between uniform LPS and hidden quantum Markov models (HQMMs) established in this paper, the HQMM learning algorithm from Adhikary et al. (2020), 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 stochastic process models such as hidden Markov models (HMMs) and PSRs (Hsu et al., 2012; Siddiqi et al., 2010; Bailly et al., 2009) could also be adapted to learn uniform LPS and uniform MPS models. 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.

Finally, one of our key motivations is to simply provide a means of translating between similar models developed in different fields. While prior works (Glasser et al., 2019; Kliesch et al., 2014; Critch and Morton, 2014) have noted similarities between tensor networks, stochastic processes and weighted automata, many formal and explicit connections are still lacking, especially in the context of model expressiveness. It is still difficult for practitioners in one field to verify that the model classes they have been working with are indeed used elsewhere, given the differences in nomenclature and domain of application; simply having a thesaurus to rigorously translate between fields can be quite valuable. Such a thesaurus is particularly timely given the growing popularity of tensor networks in machine learning. We hope that the connections presented here will help bring together complementary advances occurring in these various fields.

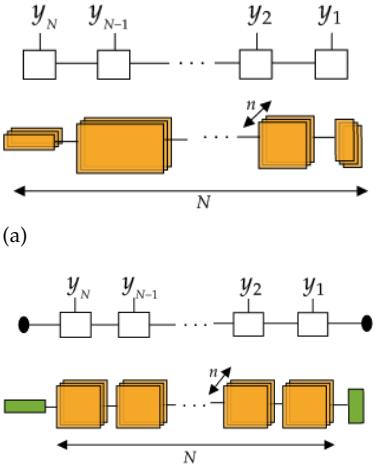


Figure 7.1: **Matrix Product States and Uniform Matrix Product States** Tensor network diagrams (top) and illustrative representations (bottom) of MPS (a) and uMPS (b).

Outline We begin with background on uniform matrix product states in Section 7.1, and show that they are equivalent to OOMs in the non-terminating limit of infinitely long sequences. In Section 7.2, we summarize known results equating non-terminating non-negative uniform MPS with HMMs, and in Section 7.3, we show that non-terminating uniform Born machines are equivalent to NOOMs. Finally, in Section 7.4, we similarly demonstrate equivalence between HQMMs and non-terminating uniform locally purified states.

Notation We use the same mathematical notation used in the earlier Chapters as described in Section 3.1. Additionally, we use tensor

network diagrams to illustrate various model classes. We refer the reader to Biamonte and Bergholm (2017) for an introduction to tensor network diagrams, and the theory of tensor networks in general.

7.1 Uniform Matrix Product States

Matrix product states (MPS) were developed in the quantum information literature as compact representations of the joint state of multi-particle quantum systems (Perez-Garcia et al., 2006; Klümper et al., 1993; Fannes et al., 1992). As discussed in Chapter 3, an n-dimensional quantum particle is characterized by a belief state enumerating the probabilities of it being in one of n basis states. A system of N such particles constitutes a joint probability tensor with n^N entries, which scales exponentially with the number of particles. An MPS representation decomposes this N-mode joint probability tensor into a series of N three-dimensional tensor cores (Figure 7.1(a)), resulting in $O(nN)$ parameters instead.

The MPS decomposition can also be equivalently applied to joint distributions over sequences of observations. Given a set of n discrete observations $\mathcal{O}$, an MPS encodes the probability of an N-length sequence of observations $y \in \mathcal{O}$ as follows

$$P(y_1, \ldots, y_N) \propto \text{MPS}_{y_1,\ldots,y_N} = \mathbf{A}^{[N],y_N}\mathbf{A}^{[N-1],y_{N-1}}\ldots\mathbf{A}^{[2],y_2}\mathbf{A}^{[1],y_1} \quad (7.1)$$

Here, each $\mathbf{A}^{[i]}$ is a three-mode tensor formed by stacking n matrix slices $\mathbf{A}^{[i],y_i}$ (one for every observation) of dimensions $D_{i+1} \times D_i$; the maximal value of D_i is the bond-dimension of the MPS (Glasser et al., 2019). We use the convention of *open boundary conditions*[1] and set $D_0 = D_N = 1$, i.e., the boundary cores have *vector* slices.

[1] The MPS decomposition with open boundary conditions is also known as the tensor-train (TT) decomposition (Oseledets, 2011). An alternate convention, *periodic boundary conditions*, sets $D_0 = D_N > 1$ and evaluates the product of matrices using the trace operation. Such an MPS is equivalent to a tensor *ring* decomposition (Mickelin and Karaman, 2018)

Uniform Matrix Product States As illustrated in Figure 7.1(a), the tensor cores of an MPS need not be the same, which means that the model must be defined with respect to a fixed sequence length. If we instead restrict all cores of the MPS to be identical, we can model arbitrarily long sequences simply by repeating the same core as many times as needed. We restrict our analysis to such stationary or *uniform* MPS (uMPS) models (Figure 7.1(b)), which encode the probability of sequences $\{y_1, \ldots, y_N\}$ as follows

$$P(y_1, \ldots, y_N) \propto \text{uMPS}_{y_1,\ldots,y_N} = \boldsymbol{\sigma}^\dagger \mathbf{A}^{y_N}\mathbf{A}^{y_{N-1}}\ldots\mathbf{A}^{y_2}\mathbf{A}^{y_1}\vec{\boldsymbol{\rho}}_0 \quad (7.2)$$

Note that we have introduced two *fixed* boundary vectors $\boldsymbol{\sigma}$ and $\vec{\boldsymbol{\rho}}_0$ that are independent of observations; a natural choice to ensure that all observation-dependent cores are identical. We can interpret these fixed boundaries $\boldsymbol{\sigma}^\dagger$ and $\vec{\boldsymbol{\rho}}_0$ as the *evaluation functional* and the

initial latent state of the uMPS respectively. Using this interpretation of an initial latent state evolving over time, we compute conditional probabilities of observations by marginalizing all future observations as follows

$$P(y_t|y_{1:t-1}) = \frac{\sum_{y_N,\ldots,y_{t+1}} P(y_N,\ldots,y_{t+1},y_t,y_{t-1},\ldots y_1)}{\sum_{y_N,\ldots,y_t} P(y_N,\ldots,y_t,y_{t-1},\ldots,y_1)} = \frac{\boldsymbol{\sigma}^\dagger \left(\sum_y \mathbf{A}_y\right)^{N-(t+1)} \mathbf{A}_{y_t} \ldots \mathbf{A}_1 \vec{\boldsymbol{\rho}}_0}{\boldsymbol{\sigma}^\dagger \left(\sum_y \mathbf{A}_y\right)^{N-t} \mathbf{A}_{y_{t-1}} \ldots \mathbf{A}_1 \vec{\boldsymbol{\rho}}_0} \tag{7.3}$$

The Non-Terminating Limit

Note that in Equation 7.3, we repeatedly apply the same matrix $\sum_y \mathbf{A}_y$, called the *transfer operator*, to the evaluation functional $\boldsymbol{\sigma}^\dagger$. In the limit where $(N-t) \to \infty$, this 'power iteration' converges to an *effective* evaluation functional $\boldsymbol{\sigma}_*^\dagger$, namely the top eigenvector of the transfer operator i.e. $\boldsymbol{\sigma}^\dagger \left(\sum_y \mathbf{A}_y\right)^{k\to\infty} = \boldsymbol{\sigma}^{*\dagger}$. In this limit[2], Equation 7.3 can be rewritten as the following simple recursive state update

$$\vec{\boldsymbol{\rho}}_t = \vec{\boldsymbol{\rho}}_{y_t|y_{1:t-1}} = \frac{\mathbf{A}_{y_t} \ldots \mathbf{A}_1 \vec{\boldsymbol{\rho}}_0}{\boldsymbol{\sigma}_*^\dagger \mathbf{A}_{y_{t-1}} \ldots \mathbf{A}_1 \vec{\boldsymbol{\rho}}_0} \qquad P(y_t|y_{1:t-1}) = \boldsymbol{\sigma}_*^\dagger \vec{\boldsymbol{\rho}}_t \tag{7.4}$$

Generally formulated as dynamical systems, stochastic process models tend to be described through recursive state updates such as this. To draw comparison with them, we will thus consider uMPS models in the earlier limit, which we call the *non-terminating limit*

[2] Even for finitely long sequences, the evaluation functional converges at an exponential rate such that $\|\boldsymbol{\sigma}_t - \boldsymbol{\sigma}_*\|^2 = \mathcal{O}(\exp \frac{N-t}{\xi})$, 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.

Definition 29 (Non-Terminating uMPS)**.** *A non-terminating uMPS is an infinitely long uMPS, and the probability of any sequence $P(y_1,\ldots,y_t)$ of length t can be computed by marginalizing over infinitely many future observations, i.e.,*
$P(y_1,\ldots,y_t) = \lim_{N\to\infty} \sum_{y_N} \cdots \sum_{y_{t+1}} P(y_1,\cdots,y_N)$.

While the recursive state update in Equation 7.4 is valid only in the non-terminating limit[3], it remains valid even outside this limit in the special case where the evaluation functional $\boldsymbol{\sigma}$ is a priori set to be the fixed point of the transfer operator, i.e., $\boldsymbol{\sigma}^\dagger \sum_y \mathbf{A}_y = \boldsymbol{\sigma}^\dagger$. As we now discuss, such a uMPS is equivalent to observable operator models (OOMs), equivalently uncontrolled predictive state representations (PSRs) or stochastic weighted automata[4].

[3] The non-terminating limit is analogous to the common practice of 'burning-in' the first few observations in a stochastic process model to allow the system to reach a stationary state; here, we apply a 'burn-out' to the end of sequences.

[4] A stochastic weighted automata model is simply any weighted automaton model that computes a probability distribution. See Adhikary et al. (2021b) for further details.

Non-Terminating uMPS are OOMs In Chapter 5, we have discussed OOMs or PSRs in some detail. As a reminder, an n-dimensional OOM is a stochastic process defined over a set of discrete observations $\mathcal{O}$ represented by a tuple $(\mathbb{C}^n, \boldsymbol{\sigma}, \{\boldsymbol{\tau}_y\}_{y\in\mathcal{O}}, \mathbf{x}_0)$. The initial state

vector $\mathbf{x}_0 \in \mathbb{C}^n$ is normalized with respect to the linear evaluation functional $\boldsymbol{\sigma}^\dagger$, i.e., $\boldsymbol{\sigma}^\dagger \mathbf{x}_0 = 1$, and the set of *observable operators* $\boldsymbol{\tau}_y$ have normalized marginals over observations $\boldsymbol{\sigma}^\dagger \sum_y \boldsymbol{\tau}_y = \boldsymbol{\sigma}^\dagger$. Finally, we require that the probability of arbitrary length sequences $y_1, \ldots, y_N \in \mathcal{O}^{\otimes N}$, computed as $\boldsymbol{\sigma}^\dagger \boldsymbol{\tau}_{y_N}, \ldots, \boldsymbol{\tau}_{y_1} \mathbf{x}_0$, must always be non-negative.

Given the constraint over the observable operators, we see that an OOM is equivalent to a (non-terminating) uMPS where the evaluation functional is defined to be the fixed point of its transfer operator $\sum_y \boldsymbol{\tau}_y$. We thus obtain the following equivalence result

Theorem 11 (**uMPS = OOM** *in the non-terminating limit*)**.** *Non-terminating uniform matrix product states are equivalent to uncontrolled predictive state representations.*

Proof. Consider a uMPS $\{\boldsymbol{\sigma}, \{\boldsymbol{\tau}_y\}_y, \vec{\boldsymbol{\rho}}_0\}$ 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

$$\begin{aligned} P(y_t \mid y_{1:t}) &= \frac{\boldsymbol{\sigma}^\dagger \boldsymbol{\tau}^{N-t} \boldsymbol{\tau}_{y_t} \left(\boldsymbol{\tau}_{y_{t-1}} \cdots \boldsymbol{\tau}_{y_1} \vec{\boldsymbol{\rho}}_0 \right)}{\boldsymbol{\sigma}^\dagger \boldsymbol{\tau}^{N-t} \boldsymbol{\tau} \left(\boldsymbol{\tau}_{y_{t-1}} \cdots \boldsymbol{\tau}_{y_1} \vec{\boldsymbol{\rho}}_0 \right)} \\ &= \frac{\boldsymbol{\sigma}_t^\dagger \boldsymbol{\tau}_{y_t} \left(\boldsymbol{\tau}_{y_{t-1}} \cdots \boldsymbol{\tau}_{y_1} \vec{\boldsymbol{\rho}}_0 \right)}{\boldsymbol{\sigma}_t^\dagger \boldsymbol{\tau} \left(\boldsymbol{\tau}_{y_{t-1}} \cdots \boldsymbol{\tau}_{y_1} \vec{\boldsymbol{\rho}}_0 \right)} \end{aligned} \tag{7.5}$$

Here, we have defined the *effective* evaluation functional $\boldsymbol{\sigma}_t = \left(\boldsymbol{\tau}^{N-t}\right)^\dagger \boldsymbol{\sigma}$ 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. 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 OOM 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).
If the top eigenvalue of $\boldsymbol{\tau}^\dagger$ is $\lambda_* = 1$, $\boldsymbol{\sigma}_t$ will eventually converge to $\boldsymbol{\sigma}_*$, 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 7.5 then reduces to that of a PSR with evaluation functional $\boldsymbol{\sigma}_*$. Note that in a PSRs, the $\boldsymbol{\sigma}$ is chosen precisely to be the fixed point of the adjoint transfer operator via the normalization requirement $\boldsymbol{\sigma}^\dagger \boldsymbol{\tau} = \boldsymbol{\sigma}^\dagger$, forcing $\lambda_* = 1$.
If $\lambda_* \neq 1$ for a uMPS model, we can simply rescale our matrices $\boldsymbol{\tau}$ to obtain a properly normalized transfer operator, by replacing $\boldsymbol{\tau}_y$ with $\boldsymbol{\tau}_y / \lambda_*$. Making this substitution in Equation 7.5, we see that the numerator and denomenator 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 $\boldsymbol{\sigma}_*$, for $\boldsymbol{\sigma}_*$ the fixed point of $\boldsymbol{\tau}^\dagger$. □

The Negative Probability Problem for uMPS In Section 5.2, we discussed that while OOMs are a highly expressive model class, they suffer from the drawback of the negative probability problem (NPP). Namely, it is undecidable whether a candidate OOM will return negative probabilities for some sequence. Since uMPS are equivalent to OOMs, it is unsurprising that they also suffer from the same problem. In fact, the NPP has been discovered somewhat independently within the quantum-information literature in the context of MPS-like tensor networks; it is undecidable whether a general matrix product operator describes a valid quantum state (Kliesch et al., 2014).

We have already seen how HMMs, NOOMs, and HQMMs can be viewed as constrained variants of general OOMs that avoid the NPP by design. Similar variants have been proposed in the MPS literature as well. We now describe these models, and discuss how they relate to corresponding OOM variants.

7.2 *Non-Negative uMPS and HMMs*

A non-negative MPS[5] is simply an MPS with non-negative model parameters. As noted by several authors (Kliesch et al., 2014; Critch and Morton, 2014), non-negative MPS are equivalent to hidden Markov models (HMMs). An n-dimensional HMM is a stochastic process

[5] Non-negative uMPS are equivalent to **probabilistic automata** (PA) from formal language theory (Denis and Esposito, 2008), 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)

generally defined by the tuple $(\mathbb{R}^n, \mathbf{A}, \mathbf{C}, \mathbf{x}_0)$, where the initial state vector $\mathbf{x}_0$ has unit 1-norm, and the transition and emission matrices $\mathbf{A}$ and $\mathbf{C}$ are column stochastic. Starting with a normalized initial state $\mathbf{x}_0$, we compute the latent state $\mathbf{x}_t$ conditioned on having observed y_t at time t using the following recursive state update

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

where $\text{diag}(\mathbf{C}_{(y,:)})$ is a diagonal matrix with the y-th row of the emission matrix $\mathbf{C}$ along its diagonal, and $\mathbf{T}_y$ are observable operators defined as $\mathbf{T}_y = \text{diag}(\mathbf{C}_{(y,:)})\mathbf{A}$ (Jaeger, 2000).

Expressiveness of non-negative uMPS Since non-negative uMPS are equivalent to HMMs, they share the same expressiveness as HMMs. Specifically, we have discussed how the set of valid HMM states defines a polyhedral cone. A clear example of the limited expressiveness of HMMs is that processes with oscillatory state-space behavior cannot be modeled exactly with a finite dimensional HMM. Thus same restriction thus also holds for non-negative uMPS models. Since the non-negative operators of non-negative uMPS have real eigenvalues – similar to HMMs – they cannot induce oscillations in the latent state. While such oscillatory behavior may not be the only class of processes that separate these model classes, they are sufficient to demonstrate that these models have limited expressiveness compared to OOMs.

Given the limited expressiveness of non-negative uMPS, it is desirable to search for alternative constraints on uMPS that are less stringent than non-negativity. We now discuss a few proposals which allow negative (or complex valued) parameters but compute probabilities via quadratic functions, forcing them to be non-negative. We will show that these models are equivalent to the similarly motivated NOOMs, and that they unfortunately fail to capture all HMMs.

7.3 Uniform Born Machines & NOOMs

Uniform Born Machines Born Machines (Han et al., 2018) are a popular class of quantum tensor networks that propose a simple solution to the NPP - simply wrap the MPS inside a quadratic function to compute probabilities. This is, in fact, one of the key postulates of quantum mechanics, called the *Born rule*, used to compute the probabilities of quantum measurements (Nielsen and Chuang, 2011). As with uMPS, we focus on *uniform* Born machines (uBMs), which

compute sequence probabilities as follows

$$
\begin{aligned}
P(y_1, \ldots, y_N) \propto \mathrm{uBM}_{y_1,\ldots,y_N} &= \left| \boldsymbol{\alpha}^\dagger \mathbf{A}^{y_N} \ldots \mathbf{A}^{y_1} \boldsymbol{\omega}_0 \right|^2 \\
&= \boldsymbol{\alpha}^\dagger \mathbf{A}^{y_N} \ldots \mathbf{A}^{y_1} \boldsymbol{\omega}_0 \, \boldsymbol{\omega}_0^\dagger (\mathbf{A}^{y_1})^\dagger \ldots (\mathbf{A}^{y_N})^\dagger \boldsymbol{\alpha} \\
&= \boldsymbol{\sigma}^\dagger \, \boldsymbol{\tau}_{y_N} \, \ldots \, \boldsymbol{\tau}_{y_1} \, \vec{\boldsymbol{\rho}}_0
\end{aligned}
\tag{7.6}
$$

where $\boldsymbol{\tau}_y = \overline{\mathbf{A}^y} \otimes \mathbf{A}^y$, $\vec{\boldsymbol{\rho}}_0 = \overline{\boldsymbol{\omega}_0} \otimes \boldsymbol{\omega}_0$, and $\boldsymbol{\sigma} = \bar{\boldsymbol{\alpha}} \otimes \boldsymbol{\alpha}$. It is clear that uBMs are a special kind of uMPS/OOM with the additional constraint that the observable operators $\boldsymbol{\tau}_y$, the evaluation functional $\boldsymbol{\sigma}$, and the initial state $\vec{\boldsymbol{\rho}}_0$ all have unit Kraus-rank [6].

[6] An operator has Kraus rank r if it has a Schmidt decomposition $\boldsymbol{\tau} = \sum_{i=1}^{r} \overline{\mathbf{X}}_i \otimes \mathbf{X}_i$ with no more than r terms.

Non-Terminating uBMs are NOOMs We recall that the notion of using quadratic evaluations to avoid negative probabilities was also the central idea behind formulating NOOMs[7] (Zhao and Jaeger, 2010b). Recall that an n-dimensional NOOM defined over a set of discrete observations $\mathcal{O}$ is characterized by a tuple $(\mathbb{C}^n, \{\boldsymbol{\phi}_y\}_{y\in\mathcal{O}}, \boldsymbol{\psi}_0)$. The initial state $\boldsymbol{\psi}_0 \in \mathbb{C}^n$ has unit 2-norm, i.e., $\|\boldsymbol{\psi}_0\|_2^2 = 1$, and the operators $\boldsymbol{\phi}_y \in \mathbb{C}^{n\times n}$ satisfy $\sum_y \boldsymbol{\phi}_y^\dagger \boldsymbol{\phi}_y = \mathbb{I}$. Sequence probabilities are computed as follows

[7] NOOMs are known as quadratic weighted automata (QWA) in the formal languages literature (Bailly, 2011).

$$
\begin{aligned}
P(y_1, \ldots, y_N) = \mathrm{NOOM}_{y_1,\ldots,y_N} &= \left|\left| \boldsymbol{\phi}_{y_N} \, \cdots \, \boldsymbol{\phi}_{y_1} \boldsymbol{\psi}_0 \right|\right|_2^2 \\
&= \mathrm{tr}(\boldsymbol{\phi}_{y_N} \, \cdots \, \boldsymbol{\phi}_{y_1} \boldsymbol{\psi}_0 \boldsymbol{\psi}_0^\dagger (\boldsymbol{\phi}_{y_1})^\dagger \, \cdots \, (\boldsymbol{\phi}_{y_N})^\dagger) \\
&= \vec{\mathbb{I}}^T \boldsymbol{\tau}_{y_N} \, \cdots \, \boldsymbol{\tau}_{y_1} \, \vec{\boldsymbol{\rho}}_0
\end{aligned}
$$

We see that NOOMs, like uBMs, are also a kind of PSR/uMPs with observable operators $\boldsymbol{\tau}_y = \overline{\boldsymbol{\phi}}_y \otimes \boldsymbol{\phi}_y$, initial state $\vec{\boldsymbol{\rho}}_0 = \mathrm{vec}(\overline{\boldsymbol{\psi}_0} \boldsymbol{\psi}_0^\dagger) = \overline{\boldsymbol{\psi}}_0 \otimes \boldsymbol{\psi}_0$, and evaluation functional $\vec{\mathbb{I}}^T$

Similar to uBMs, the operators and the initial state of NOOMs have unit Kraus-rank, but the evaluation functional $\vec{\mathbb{I}}^T$ has full Kraus-rank since it the vectorization of the full rank identity matrix. However, as we discussed in Section 7.1, the exact choice of the evaluation functional does not matter in the non-terminating limit; for infinitely long sequences, it will converge to the fixed point of its transfer operator. It can be shown that whatever the fixed point of a non-terminating uBM may be, we can always find a similarity transform to an equivalent model where the transfer operator has the NOOM evaluation functional as its fixed point (Adhikary et al., 2021b). This leads us to the following theorem

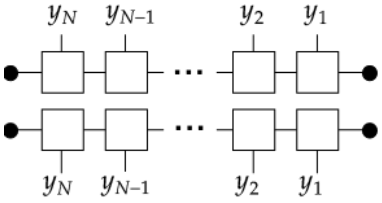


Figure 7.2: **Tensor network diagram of a uBM**

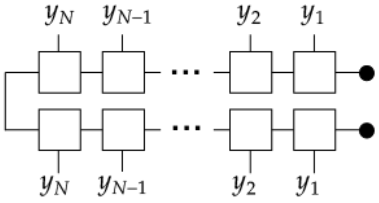


Figure 7.3: **Tensor network diagram of a NOOM**

Theorem 12 (**NOOMs = uBMs** *in the non-terminating limit*)**.** *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 that NOOM transfer operators are required to be trace-preserving. While uBMs can have an arbitrary Kraus-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 Kraus-rank 1. Therefore, we 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 (Nielsen and Chuang, 2011). With this fact demonstrated, the same argument used in the proof of Theorem 11 to prove convergence of the effective functional of a uMPS to the fixed point $\boldsymbol{\sigma}_*^\dagger$ can be applied here. This has the effect of replacing the original Kraus-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 that the two eigenvalues of $\boldsymbol{\tau}$ with greatest magnitude, λ_* and λ_2, satisfy $|\lambda_*| > |\lambda_2|$. This is similar to the assumption made in proving Theorem 11, 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.

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. 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 $\boldsymbol{\sigma}_*$ 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:

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

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) $\boldsymbol{\tau}^{\dagger}\boldsymbol{\sigma}_* = \boldsymbol{\sigma}_*$.
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 sequence limit. □

Expressiveness of uBMs By equating non-terminating uBMS to NOOMs, we can now use the expressiveness relationships established for NOOMs in Section 5.3 directly to analyze the expressiveness of uBMS. Specifically, does the quadratic evaluation strategy of uBMS provide greater expressiveness than non-negative uMPS? Glasser et al. (2019) tackle this question for the non-uniform case, showing that there exist finite non-uniform BMs that cannot be modeled by finite non-uniform HMMs; they conjecture that the reverse is also true. We are able to confirm this conjecture for the uniform case as follows:

Theorem 13 (uBM $\not\subseteq$ HMM ; HMM $\not\subseteq$ uBM)**.** *There exist finite-dimensional non-terminating uniform Born machines that have no equivalent finite-dimensional hidden Markov models, and vice-versa.*

Proof. For the first relationship uBM $\not\subseteq$ HMM, note that we already know that NOOM $\not\subseteq$ HMM. Zhao and Jaeger (2010b) demonstrate this by designing the NOOM probability clock model, where the latent state (and therefore conditional probabilities) exhibit oscillatory behavior. The negative entries in NOOM operators allow their top eigenvalues to be complex valued,

which allow for such oscillatory behavior. On the other hand, HMMs with non-negative operators with real eigenvalues cannot produce such oscillations. From Theorem 12, we know that a non-terminating uBM with the same operators as the probability clock NOOM will generate identical probabilities, and thus produce the same dynamics as the probability clock NOOM. These are then instances of uBMs that cannot be modeled by a finite dimensional HMM.
For the second relationship HMM $\not\subseteq$ uBM, note that Theorem 5 in Chapter 5 tells us that there exist HMMs that cannot be modeled by finite dimensional NOOMs. The same HMMs cannot be modeled by non-terminating finite dimensional uBMs, as these are equivalent to NOOMs. □

Thus, the quadratic evaluation trick of NOOMs and uBMs results in a relatively restrictive model class that cannot model all finite HMMs. We now look an alternative set of tensor network model constraints that provide additional expressiveness.

7.4 *Locally Purified States & HQMMs*

Locally purified states (LPS) were originally proposed as tensor-network models for quantum simulation, and were designed with the explicit intention of avoiding negative probabilities. They also have a direct correspondence to local quantum circuits with ancillary qubits (Glasser et al., 2019), and can thus be used to implement probabilistic models directly on quantum computers. Glasser et al. (2019) showed that non-uniform LPS are strictly more expressive than non-uniform HMMs. As with all other tensor networks, we instead focus on *uniform* LPS or uLPS.

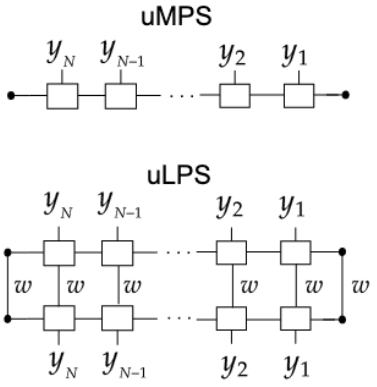


Figure 7.4: **Tensor network diagrams of uLPS and uMPS**

Compared to the general uMPS, a uLPS introduces an additional mode for each tensor core, called the "purification dimension" (Glasser et al., 2019). As illustrated in Figure 7.4, we can view uLPS as a decomposition of a uMPS along this purification dimension. Sequence probabilities are computed as follows

$$P(y_1, \ldots, y_N) \propto \text{uLPS}_{y_1,\ldots,y_N} = \left[\sum_w \overline{\mathbf{K}}^T_{w,L} \otimes \mathbf{K}^T_{w,L}\right] \left[\sum_w \overline{\mathbf{K}}_{w,y_N} \otimes \mathbf{K}_{w,y_N}\right] \cdots \left[\sum_w \overline{\mathbf{K}}_{w,y_1} \otimes \mathbf{K}_{w,y_1}\right] \left[\sum_w \overline{\mathbf{K}}_{w,R} \otimes \mathbf{K}_{w,R}\right] \tag{7.7}$$

where $\sum_w \overline{\mathbf{K}}_{w,y_i} \otimes \mathbf{K}_{w,y_i}$ are matrx slices of the tensor-cores, $\sum_{w,L} \overline{\mathbf{K}}^T_{w,L} \otimes \mathbf{K}^T_{w,L}$ is the evaluation functional, and $\sum_w \overline{\mathbf{K}}_{w,R} \otimes \mathbf{K}_{w,R}$ is the initial state vector. The sum over the w indices contracts over the

purification-dimension, and the number of summands corresponds to the model's 'puri-rank' (Glasser et al., 2019). We now show that uLPS tensor-networks are equivalent to HQMMs.

Non-terminating uLPS are HQMMs Recall the Liouville representation HQMMs in Definition 20 from Chapter 5: the initial state $\vec{\boldsymbol{\rho}}_0$ is a vectorized unit-trace Hermitian PSD matrix of arbitrary rank. The operators $\mathbf{L}_y$ satisfy trace preservation (TP), i.e, $\vec{\mathbb{I}}^T \left(\sum_{y\in\mathcal{O}}\right) \mathbf{L}_y) = \mathbb{I}^T$, and their corresponding Choi matrices are HP and CP such that $\{\mathbf{C}_y\}_{y\in\mathcal{O}}$ satisfy $\mathbf{C}_y \succeq 0$ and $\mathbf{C}_y = \mathbf{C}_y^\dagger$. State updates and probabilities of sequences are computed as follows

$$\vec{\boldsymbol{\rho}}_t|y = \frac{\mathbf{L}_y\vec{\boldsymbol{\rho}}_{t-1}}{\vec{\mathbb{I}}^T\mathbf{L}_y\vec{\boldsymbol{\rho}}_{t-1}} \qquad P(y_1,\ldots,y_t) = \vec{\mathbb{I}}^T\mathbf{L}_{y_t}\ \ldots\ \mathbf{L}_{y_1}\vec{\boldsymbol{\rho}}_0 \tag{7.8}$$

We can recover the original Kraus matrix formulation from L-HQMMs via a simple (inverse) vectorization as shown below

$$\begin{aligned} P(y_1,\ldots,y_N) &= \operatorname{tr}\left(\sum_w \mathbf{K}_{w,y_N}\cdots\sum_w \mathbf{K}_{w,y_1}\boldsymbol{\rho}\mathbf{K}_{w,y_1}^\dagger\cdots\mathbf{K}_{w,y_N}^\dagger\right) \\ &= \vec{\mathbb{I}}^T \underbrace{\left(\sum_w \overline{\mathbf{K}}_{w,y_N}\otimes\mathbf{K}_{w,y_N}\right)}_{\mathbf{L}_{y_N}}\cdots\underbrace{\left(\sum_w \overline{\mathbf{K}}_{w,y_1}\otimes\mathbf{K}_{w,y_1}\right)}_{\mathbf{L}_{y_1}}\vec{\boldsymbol{\rho}}_0 \end{aligned}$$

Note that this is identical to the probability computation for a uLPS in Equation 7.7, with the only exception that an HQMM has a fixed evaluation functional $\vec{\mathbb{I}}^T$. But recall that this difference at the boundary does not matter in the non-terminating limit. Following essentially the same proof strategy as with NOOMs-uBMs, we obtain the following result

Theorem 14 (uLPS = HQMM)**.** *Non-terminating uniform locally purified states are equivalent to hidden quantum Markov models.*

Proof. We follow the same approach as in Theorem 12. Uniform LPS models and HQMMs differ only in their evaluation functionals, and 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.
As discussed in Theorem 12, 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. □

7.5 Tensor Networks with Control

The expressiveness relationships established in this chapter also translate to the controlled setting. Just as we defined IO-HQMMs from HQMMs, we can define input-output uLPSs and input-output uMPSs. The non-terminating variants of these models would then being equivalent to IO-HQMMs and IO-OOMs/PSRs respectively. When viewed as a tensor network in Figure 7.5, we also note that IO-HQMMs and uniform IO-LPSs have the same structure as matrix product operators (Chan et al., 2016; Murg et al., 2008) which can be viewed as MPSs with an additional open index at each core.

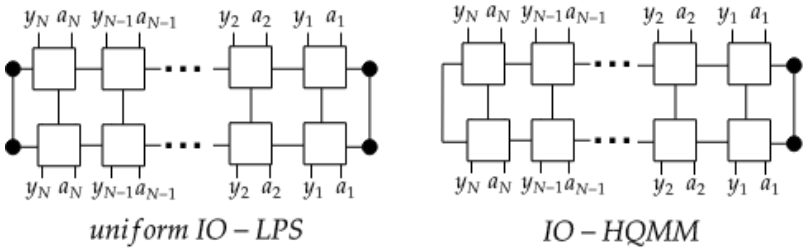


Figure 7.5: **Tensor network diagrams for IO-HQMM and uniform IO-LPS**

7.6 Conclusion

We established formal connections between various probabilistic models of stochastic linear processes proposed in machine learning, quantum tensor networks, and formal language theory. While various expressiveness relationships had been established for non-uniform quantum tensor networks, the same were not available for the *uniform* case where parameters are shared across time steps. We focused on this uniform setting and derived new expressiveness relationships between MPS and its variants including non-negative MPS and Born machines, and linear sequential systems including observable operator models or predictive state representations, and hidden quantum Markov models. Using these equivalence relationships, we are able to concisely visualize the hierarchy of expressiveness between the models we have considered in Figure 7.6. These relationships can also be qualitatively inferred by viewing these models as tensor networks as in Figure 7.6.

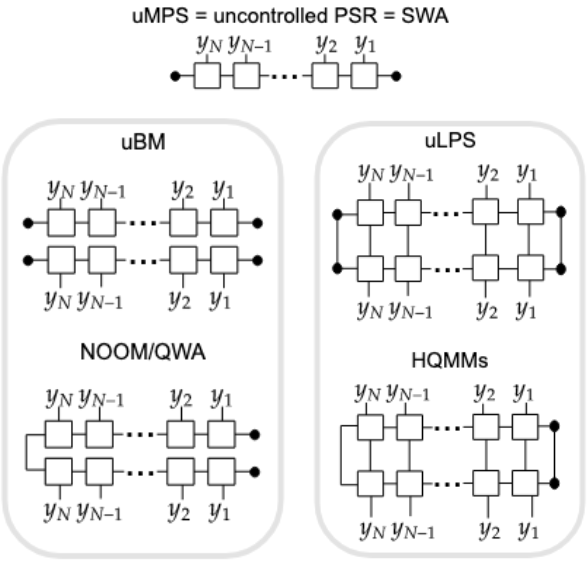


Figure 7.6: **All Models Considered** Tensor network diagrams for the models considered.

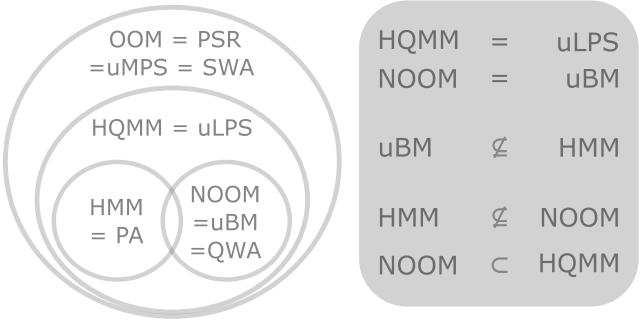


Figure 7.7: **Hierarchy of expressiveness of quantum tensor networks, linear sequential systems, and weighted automata** Relative expressiveness of various models considered in this chapter. The gray box indicates new relationships established.

We anticipate that the equivalences we have developed will serve as a thesaurus to be able to translate between similar models proposed in different communities, and enable the transfer of insights and algorithms across these domains.

Part III

Metric Informed Sampling

8
Kernel Herding over Riemannian Manifolds

Abstract *Kernel herding is a deterministic sampling algorithm designed to draw 'super samples' from probability distributions when provided with their kernel mean embeddings in a reproducing kernel Hilbert space (RKHS). Empirical expectations of functions in the RKHS formed using these super samples tend to converge even faster than random sampling from the true distribution itself. Standard implementations of kernel herding have been restricted to sampling over flat Euclidean spaces, which is not ideal for applications such as robotics where more general Riemannian manifolds may be appropriate. We propose to adapt kernel herding to Riemannian manifolds by (1) using geometry-aware kernels that incorporate the appropriate distance metric for the manifold and (2) using Riemannian optimization to constrain herded samples to lie on the manifold.*

We evaluate our approach on problems involving various manifolds commonly used in robotics including the SO(3) manifold of rotation matrices, the spherical manifold used to encode unit quaternions, and the manifold of symmetric positive definite matrices. We demonstrate that our approach outperforms existing alternatives on the task of resampling from empirical distributions of weighted particles, a problem encountered in applications such as particle filtering. We also demonstrate how Riemannian kernel herding can be used as part of the kernel recursive approximate Bayesian computation algorithm to estimate parameters of black-box simulators that lie on Riemannian manifolds, including inertia matrices of an Adroit robot hand simulator. Our results confirm that exploiting geometric information through our approach to kernel herding yields better results than alternatives including standard kernel herding with heuristic projections.

8.1 Introduction

Kernel Herding Kernel mean embeddings are powerful statistical tools that enable learning and inference in potentially infinite-dimensional feature spaces. By representing probability distributions as elements of reproducing kernel Hilbert spaces (RKHS), kernel mean embeddings allow us to apply various kernel-based algorithms directly on distributions (Song et al., 2013; Muandet et al., 2017). While there are many tools available to sample from probability

density functions, they are not directly applicable to kernel mean embeddings. *Kernel herding* is a deterministic algorithm that allows us to draw samples from distributions when they are represented as kernel mean embeddings (Chen et al., 2010). This is particularly useful for methods that operate in an RKHS, including many algorithms designed for robotics problems such as filtering (Kanagawa et al., 2016; Lacoste-Julien et al., 2015), simulator parameter estimation (Kajihara et al., 2018; Kisamori et al., 2020), and likelihood-free Bayesian inference (Hsu and Ramos, 2019; Kajihara et al., 2018). Adapting kernel herding to Riemannian manifolds also extends the applicability of these algorithms to problems defined over Riemannian manifolds.

Kernel herding also provides an efficient means to summarize large datasets using a smaller, weighted subset that preserves distributional properties in an RKHS. Most kernel based algorithms where one might encounter kernel mean embeddings tend to decouple algorithmic and sample complexities from the dimensionality of feature spaces; instead of scaling with feature dimensions, these methods generally scale with the number of data points. This highlights the utility of being able to sample a smaller set of representative data points – also called a core-set – instead of evaluating kernels on the entire dataset. In the absence of access to the data generating distribution, kernel herding can be useful to draw a core-set of samples from the empirical distribution of the existing dataset, reducing both the memory footprint and computational time of the underlying algorithms.

Riemannian Manifolds Current applications of kernel herding have been restricted to sampling over flat Euclidean spaces (Chen et al., 2010; Kanagawa et al., 2016; Kajihara et al., 2018; Chen et al., 2018), and the standard kernel herding algorithm does not directly translate to general Riemannian manifolds encountered in many applications. For example, in robotics, quantities like stiffness and inertia are represented by symmetric positive definite (SPD) matrices, which form Riemannian manifolds when equipped with a Riemannian metric. The special orthogonal group SO(3) of rotation matrices is used to represent robot orientations and rotations. The spherical manifold of unit-vectors can also be used to encode orientations and rotations as unit quaternions. The spherical manifold is also foundational in directional statistics (Mardia and Jupp, 2009; Sra, 2018) and has a wide range of applications including topic modeling in natural language processing (Reisinger et al., 2010), protein structure modeling (Mardia et al., 2007), speaker clustering from audio data (Tang et al., 2009), etc. Exploiting the Riemannian structure of data spaces has proven to be useful in numerous applications in robotics

including model predictive control (Calinon, 2020), imitation learning (Zeestraten et al., 2017), Bayesian optimization (Jaquier et al., 2020), and so on. Our aim is to similarly incorporate the Riemannian structure of data spaces for the task of sampling from empirical distributions and kernel mean embeddings.

Kernel Herding on Riemannian Manifolds We propose to adapt kernel herding to Riemannian manifolds by (1) using geometry-aware kernels that incorporate the appropriate distance metric for the manifold, and (2) using Riemannian optimization to ensure that we only generate feasible samples that lie on the manifold. Geometry-aware kernels have been successfully applied to many problems on Riemannian manifolds including Bayesian optimization (Jaquier et al., 2020; Oh et al., 2018), spectral image classification (Honeine and Richard, 2010) and object recognition (Jayasumana et al., 2013a). Our approach is similar in spirit to that of Jaquier et al. (2020), which applies a combination of geometry-aware kernels and Riemannian optimization to Bayesian optimization. We focus on a different problem of sampling over Riemannian manifolds.

While sampling from distributions over Riemannian manifolds has been tackled in various contexts, the problem of sampling from kernel mean embeddings over Riemannian manifolds is yet to be addressed. There is significant literature dedicated to extending Markov chain Monte Carlo (MCMC) methods to Riemannian manifolds (Girolami and Calderhead, 2011; Brubaker et al., 2012; Byrne and Girolami, 2013; Diaconis et al., 2013). Utilizing the Riemannian kernelized Stein discrepancy (Liu and Zhu, 2018; Barp et al., 2022), the Riemannian Stein variational gradient descent algorithm provides a particle-based alternative that mitigates unwanted positive sample correlations observed in MCMC methods (Liu and Zhu, 2018). The same discrepancy measure may also allow a similar extension of other sampling algorithms such as Stein points (Chen et al., 2018). However, all of these methods rely on having access to (potentially unnormalized) probability density functions. In contrast, we focus on a different access model where samples are to be drawn from distributions embedded in an RKHS or empirical distributions defined via sets of weighted particles.

Outline We begin with an introduction to the standard kernel herding algorithm in Section 8.2, and then in Section 8.3, we describe our proposal of extending it to Riemannian manifolds. In Section 8.4, we evaluate our approach in drawing resamples from empirical distributions over Riemannian manifolds. Finally, in Section 8.5 we evaluate the efficacy of using Riemannian kernel herding as part of the Kernel

Recursive Approximate Bayesian Computation (KR-ABC) to estimate simulator parameters defined over Riemannian manifolds.

Notation We use bold-face capitalized letters for matrix operators (e.g. $\mathbf{A}$), bold-face lower case letters for vectors (e.g. $\mathbf{x}$) and plain non-bold symbols for scalars. We use $\mathbb{R}$ and $\mathbb{C}$ to denote the real and complex scalar fields. We use $\mathbb{E}$ to denote expectations, and $\mathcal{I}$ denotes the indicator function. The inner-product defined over a vector space $\mathcal{V}$ is represented as $\langle \boldsymbol{v}_1, \boldsymbol{v}_2 \rangle_{\mathcal{V}}$ for all $\boldsymbol{v}_1, \boldsymbol{v}_2 \in \mathcal{V}$, and we denote the norm defined with respect to this inner-product as $\|\boldsymbol{v}\|_{\mathcal{V}} = \langle \boldsymbol{v}, \boldsymbol{v} \rangle_{\mathcal{V}}$ for all $\boldsymbol{v} \in \mathcal{V}$.

Let $\mathcal{X}$ denote a data-space (or pre-image space) consisting of data points $\mathbf{x} \in \mathcal{X}$. These data points are mapped onto a feature space Φ (or image space) via a feature map $\phi : \mathcal{X} \to \Phi$. Unless otherwise stated, we assume that $\mathcal{X}$ has finite dimensions, while Φ is infinite-dimensional. Given a probability distribution P defined over a space $\mathcal{S}$, we use $\boldsymbol{s} \sim P$ to indicate that $\boldsymbol{s} \in \mathcal{S}$ was sampled from the distribution P. Feature maps ϕ can themselves be viewed as elements of infinite-dimensional spaces, regardless of the dimensionality of Φ. When viewing a feature map ϕ as such an infinite-dimensional vector, we will use the bold-face $\boldsymbol{\phi}$. We reserve $\mathcal{H}$ to denote a specific class of feature spaces known as a Hilbert space, which essentially corresponds to a feature space equipped with an inner-product.

8.2 *Kernel Herding*

Let $\boldsymbol{\mu}_P$ be the kernel-mean embedding encoding the distribution P from which we wish to draw samples. Recall[1] that a kernel mean embedding $\boldsymbol{\mu}_P$ is the representation of a probability distribution P embedded in a reproducing kernel Hilbert space (RKHS) via a kernel function k, and corresponds to the expected value of the kernel function $\mathbb{E}_{\mathbf{x} \sim P} k(\mathbf{x}, \cdot)$. Given a kernel function k and this target embedding $\boldsymbol{\mu}_P$, kernel herding sequentially constructs a new set of samples $\{\tilde{\mathbf{x}}_t\}_{t=1}^n$ by solving the following optimization problem n times (Chen et al., 2010):

[1] We provide background on kernel mean embeddings and the maximum mean discrepancy in Section 2.3

$$\begin{aligned} \tilde{\mathbf{x}}_t &= \underset{\mathbf{x}}{\operatorname{argmin}} \; \mathcal{L}_k(\, \mathbf{x}, \; \boldsymbol{\mu}_P, \; \{\mathbf{x}_i\}_{i=1}^{t-1} \,) \\ &= \underset{\mathbf{x}}{\operatorname{argmin}} - \langle k(\mathbf{x}, \cdot), \; \boldsymbol{\mu}_P \rangle + \mathcal{I}(t > 1) \sum_{j=1}^{t-1} \frac{1}{t} k(\mathbf{x}, \tilde{\mathbf{x}}_j) \end{aligned} \tag{8.1}$$

where, $\mathcal{I}(t > 1)$ is the indicator function ensuring that the second component of the objective is not applied when generating the first sample $\mathbf{x}_1$.

Minimizing the first component $-\langle k(\mathbf{x}, \cdot),\ \boldsymbol{\mu}_P \rangle$ of the objective in Equation 8.1 pushes the kernel mean embedding of the generated samples $k(\mathbf{x}_i, \cdot)$ towards the target embedding $\boldsymbol{\mu}_p$. After the first sample $\mathbf{x}_1$ has been generated, the second component of the objective applies a repulsive force pushing new samples to be dissimilar to those already generated in prior iterations. The measure of dissimilarity defining this repulsion in the second component, as with the similarity defining the attraction in the first component, is dictated by the kernel k.

In most practical applications, we do not have access to the actual kernel mean embedding $\boldsymbol{\mu}_P$, but instead have an *estimate* $\hat{\boldsymbol{\mu}}_P$ of it in the form of a set of samples $\{\mathbf{x}_i\}$ and corresponding weights $\{w_i > 0\}$ s.t. $\sum_i w_i = 1$. This estimated embedding is evaluated as $\hat{\boldsymbol{\mu}}_P = \sum_i w_i k(\mathbf{x}, \mathbf{x}_i)$, giving rise to the following kernel herding objective:

$$\tilde{\mathbf{x}}_i = \operatorname*{argmin}_{\mathbf{x}} - \sum_{i=1}^{n} w_i k(\mathbf{x}, \mathbf{x}_i) + \mathcal{I}(t > 1) \sum_{j=1}^{t-1} \frac{1}{t} k(\mathbf{x}, \tilde{\mathbf{x}}_j) \tag{8.2}$$

Note that the samples in the dataset $\{\mathbf{x}_i\}$ do not themselves have to have been drawn from P – we can adjust the weights $\{w_i\}$ so that the empirical distribution is a good approximation of the true distribution P. Of course, the degree to which the empirical distribution matches P will depend on the data points[2].

[2] For example, if all the data points $\mathbf{x}_i$ are the same, we cannot accurately approximate any distribution other than the delta distribution at $\mathbf{x}_i$.

Kernel Herding Generates Super Samples As shown in Chen et al. (2010), if $k(\mathbf{x}, \mathbf{x}) = R > 0$ for all $\mathbf{x} \in \mathcal{X}$, kernel herding generates samples that minimize the maximum mean discrepaency (MMD) (Gretton et al., 2012) defined as follows

$$\mathcal{L}_{\text{MMD}} = \left|\left| \hat{\boldsymbol{\mu}}_P - \frac{1}{n} \sum_{t=1}^{n} k(\mathbf{x}_t, \cdot) \right|\right|_{\mathcal{H}}^2$$

For bounded kernels, kernel herding decreases $\mathcal{L}_{\text{MMD}}$ at a rate of $O(n^{-1/2})$, and under additional assumptions (only guaranteed for finite-dimensional $\mathcal{H}$), this convergence rate improves to $O(n^{-1})$. Since this is faster than random sampling from P itself (Bach et al., 2012; Chen et al., 2010), herded samples are also called 'super samples'. While $\mathcal{H}$ is generally not finite dimensional, we still tend to get fast convergence in practice, even when Equation 8.2 is only solved approximately (Chen et al., 2010).

As discussed earlier, the second component in Equation 8.2 corresponds to a repulsive force that penalizes $\mathbf{x}$ that are similar – as determined by the kernel – to those already herded in prior iterations. The fast convergence of kernel herding is often attributed to this repulsive force that can induce negative auto-correlation between

the herded samples (Chen et al., 2010). Positive auto-correlation between generated samples can cause them to be tightly clustered for small n, as observed in various MCMC methods (Geyer, 2011). Other kernel-based sampling methods such as Stein variational gradient descent (Liu and Wang, 2016) and Stein points (Chen et al., 2018) also employ a similar repulsive force. The kernel herding updates in Equation 8.2 can also be interpreted as a special case of the Frank Wolfe algorithm (Bach et al., 2012).

The Need for Kernel Herding in Riemannian Manifolds Current applications of kernel herding have been restricted to sampling over flat Euclidean spaces (Chen et al., 2010; Kanagawa et al., 2016; Kajihara et al., 2018; Chen et al., 2018), and the standard kernel herding algorithm does not directly translate to general Riemannian manifolds encountered in many applications. For example, in robotics, quantities like stiffness and inertia are represented by symmetric positive definite (SPD) matrices, which form Riemannian manifolds when equipped with a Riemannian metric. The special orthogonal group SO(3) of rotation matrices is used to represent robot orientations and rotations. The spherical manifold of unit-vectors can also be used to encode orientations and rotations as unit quaternions. The spherical manifold is also foundational to directional statistics (Mardia and Jupp, 2009; Sra, 2018) and has a wide range of applications including topic modeling in natural language processing (Reisinger et al., 2010), protein structure modeling (Mardia et al., 2007), speaker clustering from audio data (Tang et al., 2009), etc.

Exploiting the Riemannian structure of data spaces has proven to be useful in numerous applications in robotics including model predictive control (Calinon, 2020), imitation learning (Zeestraten et al., 2017), Bayesian optimization (Jaquier et al., 2020), and so on. Adapting kernel herding to Riemannian manifolds also extends the applicability of these algorithms to problems defined over Riemannian manifolds.

We now discuss our approach to incorporate the Riemannian structure of data spaces for the task of sampling from empirical distributions and kernel mean embeddings via kernel herding.

8.3 Kernel Herding over Riemannian Manifolds

Adapting Kernel Herding to Riemannian Manifolds We identify two attributes of kernel herding that make it particularly amenable to be adapted to general Riemannian manifolds. Firstly, the use of kernels offers flexibility in incorporating new manifolds by swapping out the kernel. Secondly, the updates in Equation 8.2 do not

require a specific optimization approach – allowing us to easily inject Riemannian optimization techniques.

Since the kernel herding algorithm accesses data exclusively through kernel evaluations, geometric structure of the data space can be injected by picking an appropriate kernel. Such geometry-aware kernels take into account the curvature of the underlying manifold when assigning similarities between data points. Thus a kernel based approach such as kernel herding is attractive in its flexibility – we should ideally be able to swap out the kernel and keep the rest of the algorithm intact. As we now discuss, it turns out that we do need to take some additional care in ensuring the optimization used to solve Equation 8.2 is also valid for Riemannian manifolds.

Finally, the herding algorithm does not prescribe a particular method to solve Equation 8.2; it has been been solved using approaches including gradient descent (Chen et al., 2010), black-box optimization (Chen et al., 2018), and exhaustive search over a restricted search-space (Kanagawa et al., 2016). Over the years, Riemannian counterparts have been developed for many such optimization routines that optimize directly on the manifold of interest (Absil et al., 2007b; Boumal, 2023).

Thus, our approach to kernel herding over Riemannian manifolds is to (1) use kernels that incorporate the correct notion of distance for the manifold, and (2) use Riemannian optimization techniques to restrict herded samples on desired manifolds. Since the theoretical properties of kernel herding are agnostic to these two adjustments, convergence guarantees of the original algorithm carry over to our Riemannian adaptation. We now summarize the two components of our proposal to adapt kernel herding to non-Euclidean Riemannian manifolds.

Geodesic Exponential Kernels The first component in our Riemannian kernel herding scheme is to use geometry-aware kernels that assign similarity based on the curvature of the data manifold. Various geometry-aware kernels have been specifically designed for manifolds such as the Grassmann manifold (Hamm and Lee, 2008a,b), symmetric positive definite matrices (Jayasumana et al., 2013b), cylinders (Oh et al., 2018), etc. As a simpler alternative to designing problem specific kernels, we explore the use of *generic* kernels that are applicable across a wide range of domains. One such kernel is the exponential kernel defined as follows

$$k(\mathbf{x}, \mathbf{y}) = \exp\left(-\lambda\ d(\mathbf{x}, \mathbf{y})^q\right) \quad \lambda,\ q > 0 \tag{8.3}$$

where λ is the bandwidth, and $d(\mathbf{x}, \mathbf{y})$ is a distance function raised to the q-th power. When $d(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||_2$, we obtain the well-known

Euclidean Laplacian ($q = 1$) and Gaussian ($q = 2$) kernels.

Positive Definiteness of Geodesic Exponential Kernels A natural way to adapt these kernels to a general Riemannian manifold $\mathcal{M}$ is to simply set the distance function $d(\mathbf{x}, \mathbf{y})$ to be its geodesic distance $d_{\mathcal{M}}(\mathbf{x}, \mathbf{y})$ (Jayasumana et al., 2015, 2013b; Jaquier et al., 2020). However, the positivity of the kernel evaluation in Equation 8.3 does not guarantee a positive definite kernel; we require that the kernel always results in positive semi-definite kernel matrices (Muandet et al., 2017). Indeed, *geodesic exponential kernels* are, in general, not SPD (Feragen et al., 2015) and thus may not define a valid RKHS. The geodesic Gaussian kernel is SPD for all bandwidths only if the manifold is isomorphic to Euclidean space (Feragen et al., 2015). The geodesic Laplacian kernel is slightly more flexible and is SPD for all bandwidths if the geodesic distance is a conditionally negative definite metric, which is the case for manifolds including spheres and hyperbolic spaces (Feragen et al., 2015).

While geodesic exponential kernels may not be SPD for *all* bandwidths, they tend to be SPD with very high probability for bandwidths above some threshold λ_{thresh} when evaluated on finite datasets (Feragen and Hauberg, 2016). As the kernel bandwidth goes to infinity, the kernel matrix approaches the identity, which is SPD with non-negative eigenvalues. Since the minimum eigenvalue function is continuous, there exists some threshold bandwidth λ_{thresh} beyond which the kernel is always SPD. Prior works (Feragen and Hauberg, 2016; Jaquier et al., 2020) have estimated this bandwidth threshold by checking how often the kernel matrix is positive definite when evaluated on random sub-samples of data for different bandwidths. If this threshold is too large, the kernel matrix could degenerate into the identity, assigning zero similarity to non-identical samples. However, empirical results (Feragen and Hauberg, 2016; Jaquier et al., 2020), including our own, suggest that threshold bandwidths for geodesic exponential kernels tend to be small enough for the kernel to remain sufficiently discriminative. For example, in Figure 8.1, we present results of such an empirical evaluation for the SPD manifold.

From a practical standpoint, geodesic exponential kernels are very flexible. All we need to apply them on a manifold is its geodesic distance, and the empirically calculated bandwidth threshold. In our experiments, we opt for geodesic Laplacian kernels ($q = 1$ in Equation 8.3). We found this kernel to outperform the Gaussian kernel on resampling experiments over various manifolds. Moreover, the Laplacian kernel is provably SPD for the spherical manifold, an important manifold routinely encountered in many practical applications. Recently, kernels with q greater than but still close to 1 have been

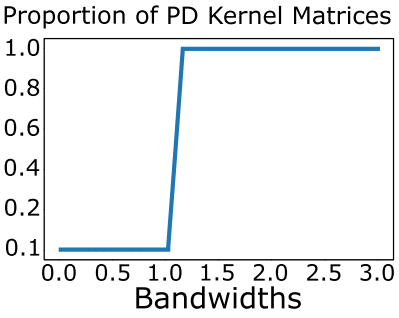


Figure 8.1: **Empirically estimating SPD threshold bandwidth for geodesic exponential kernels** For a dataset of SPD matrices drawn from a target distribution, we computed geodesic Laplacian kernels with varying bandwidths on random subsets of the data. We plot the proportion of these kernel matrices that were SPD. Beyond $\lambda_{\text{thresh}} \approx 1$ kernel matrices were almost always SPD (proportion. ≈ 1.)

shown to work well for a different problem setting over the spherical manifold of unit quaternions (Birdal et al., 2020); the authors chose to avoid the exact Laplacian kernel due to gradient instabilities in their optimization problem.

Optimization over Riemannian Manifolds The second component in our Riemannian kernel herding scheme is to swap Euclidean optimization methods used to solve Equation 8.2 with their Riemannian generalizations. Many Euclidean optimization algorithms have been generalized to Riemannian manifolds (Absil et al., 2007b; Boumal, 2023) including first-order gradient based methods like steepest descent and conjugate gradient methods (Smith, 1994), second-order methods like Newton's method and trust-region methods (Absil et al., 2007a; Smith, 1994), and black-box optimization methods such as the Nelder-Mead algorithm (Dreisigmeyer, 2007). The key insight in these generalizations is to translate motion along a straight line in flat spaces to motion along curves. For instance, consider the case of Riemannian gradient descent, which we use in our experiments. Gradient descent follows a simple recipe: step along the direction opposite to the gradient $g(\mathbf{x})$ of the objective until we arrive at an optima. In Euclidean space, this is done simply by adding $-\alpha g(\mathbf{x})$ to our current estimate $\mathbf{x}$, where α is the step-size.

However, in non-Euclidean spaces, we cannot add the gradient $g(\mathbf{x}) \in \mathcal{T}_{\mathbf{x}}\mathcal{M}$ to points on the manifold, which is generally not isomorphic to $\mathcal{T}_{\mathbf{x}}\mathcal{M}$. Thus, we must first map motion along the gradient direction onto a smooth curve on the manifold – obtaining a *curve* of steepest descent. This is achieved using a *retraction map* $\mathcal{R}_{\mathbf{x}} : \mathcal{T}_{\mathbf{x}}\mathcal{M} \to \mathcal{M}$, which corresponds to moving along a smooth curve $\gamma(t)$ on the manifold such that we start at our current estimate ($\gamma(0) = \mathbf{x}$) pointed towards the negative gradient ($\gamma'(0) = -g(\mathbf{x})$). The gradient update step is then simply computed as $\mathbf{x}_t = \mathcal{R}_{\mathbf{x}_{t-1}}(-\alpha g(\mathbf{x}_{t-1}))$. Retraction maps can be derived in closed-form for many commonly used Riemannian manifolds[3].

Since gradient descent can be slow to converge, it is generally supplemented with additional optimization techniques such as the conjugate gradient method (Hestenes and Stiefel, 1952; Dai and Yuan, 1999), momentum (Polyak, 1964; Qian, 1999a), RMSprop, Adam (Kingma and Ba, 2015), etc. These adaptive strategies combine the current gradient of the objective with gradients computed at prior iterations to form averaged descent directions. In Euclidean space, this is done simply as a weighted linear combination since all gradients are elements of the same tangent space. For general Riemannian manifolds, we must first map all gradients to be combined into a common tangent space. This is done using the manifold's *parallel transport map* $\Gamma_{\mathbf{x}\to\mathbf{y}} : \mathcal{T}_{\mathbf{x}}\mathcal{M} \to \mathcal{T}_{\mathbf{y}}\mathcal{M}$, which maps elements from one

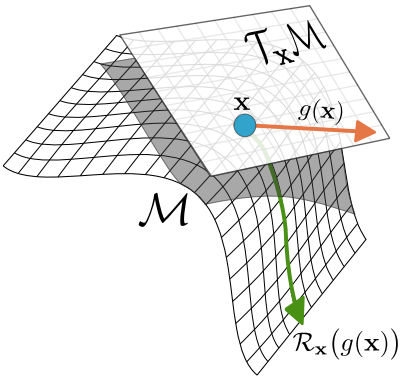


Figure 8.2: **Illustration of Retractions over Riemannian Manifolds**: The gradient $g(\mathbf{x})$ at a point $\mathbf{x}$ on a Riemannian manifold $\mathcal{M}$ is a vector on the tangent space $\mathcal{T}_{\mathbf{x}}\mathcal{M}$. Stepping along the vector $g(\mathbf{x})$ would, in general, result in going off the manifold, whereas stepping along its retraction $\mathcal{R}_{\mathbf{x}}(g(\mathbf{x}))$ follows a geodesic on the manifold.

[3] For example, we used a specific retraction map for the Stiefel manifold in our learning algorithm for HQMMs in Chapter 4

tangent space to another while preserving inner products. Bécigneul and Ganea (2019) have developed Riemannian counterparts of various adaptive gradient descent schemes including the Riemannian Adam, which is the optimization algorithm used in our experiments unless noted otherwise. This optimization scheme provides computational benefits over various second-order methods (Absil et al., 2007a; Smith, 1994), and tends to outperform vanilla gradient descent through adaptive gradient updates. Indeed, the traditional Euclidean Adam (Kingma and Ba, 2015) optimizer has been a staple across machine learning applications, and is also used in our Euclidean baseline comparisons.

We evaluate our proposal for Riemannian kernel herding on two tasks: resampling from empirical distributions and estimating parameters of black-box simulators. We now discuss these experiments in detail, beginning with the resampling task.

8.4 Resampling from Empirical Distributions

Given a closed-form expression of a probability distribution – or, more precisely, its score function in most cases – there are a multitude of options of algorithms that we can use to draw samples. However, in many applications, we may not have access to the actual distribution, but just a set of samples drawn from it. For instance, if we wish to generate additional data points to append a dataset, we will likely not have access to the density function for the data-generating distribution. Instead, we would need to somehow draw samples from the *empirical* distribution implicitly defined by the dataset.

The problem of resampling from empirical distributions can be stated as follows: given a set of non-uniformly weighted samples $\{w_i, \mathbf{x}_i\}_{i=1}^N$, we wish to generate uniformly weighted samples $\{\tilde{\mathbf{x}}_i\}_{i=1}^N$ that match the distribution of the original set. The weights $\{w_i\}$ such that $\sum_i w_i = 1$ define an empirical distribution $\eta(\mathbf{x}) = \sum_{i=1}^N w_i \mathcal{I}(\mathbf{x} - \mathbf{x}_i)$, where $\mathcal{I}$ is the indicator function.

8.4.1 Baseline Method: Optimal Transport

The baseline method for resampling over Riemannian manifolds against which we compare our approach builds off of the ideas of sampling importance-resampling. Wang and Solo (2020) propose an algorithm for particle filtering over Lie groups, using methods from optimal transport (OT) theory to draw resamples from a weighted set of particles from Lie groups[4] including SO(3). While the original resampling approach in Wang and Solo (2020) was specifically designed for Lie groups, we adapt the general strategy to Rieman-

[4] A Lie group is a differentiable manifold with additional group structure.

nian manifolds with known geodesic distance functions. Specifically, the particle filter outlined by Wang and Solo (2020) consists of two components: a method of simulating stochastic differential equations on Lie groups via the Cayley transform, and a resampling technique to overcome particle degenerancy. We focus on the latter resampling algorithm as it relates directly to our approach. We refer to this approach as the *OT Resampling* technique, and describe it in detail below.

Since the OT technique builds off of the standard particle filtering algorithm, we begin with an introduction to particle filters. We refer the reader to Doucet et al. (2009) for a detailed review of the topics covered here.

Particle Filtering Particle filtering, also known as Sequential Monte Carlo (SMC), is a powerful class of algorithms used in the analysis and control of dynamical systems. These algorithms solve the *filtering* problem of estimating the posterior distribution of the latent state of state-space models such as HMMs, from a history of noisy and partial observations. Particle filters have found significant application in fields such as robotics (Thrun et al., 2005), where they are used for tasks like robot localization and simultaneous localization and mapping (SLAM); signal processing (Arulampalam et al., 2002), particularly for target tracking and object recognition, and for time-varying sequential estimation problems in domains such as finance (Johannes and Polson, 2010).

In the context of particle filtering, the objective is to estimate the latent state at time t given the sequence of observations thus far. More precisely, we wish to estimate a posterior belief over the latent states, i.e., a probability distribution calculated as

$$P(\mathbf{x}_t|y_{1:t}) = \frac{P(y_t|\mathbf{x}_t)P(\mathbf{x}_t|y_{1:t-1})}{P(y_t|y_{1:t-1})}$$

Computing the above requires calculations of integrals, the analytic solutions for which can be prohibitively expensive to compute or even estimate without making some simplifying assumptions on the distributions. For instance, if we make the familiar assumption that the underlying distributions are Gaussian, the integral calculations simplify and an analytic solution is within reach. Particle filters allow us to avoid such simplifying assumptions on the models, by estimating the posterior distribution via samples or *particles*.

Namely, if we assume that samples $\{\mathbf{x}_t\}$ have been drawn from the posterior $P(\mathbf{x}_t|y_{1:t})$, we can estimate the posterior as the following

empirical distribution:

$$P(\mathbf{x}_t|y_{1:t}) = \frac{1}{N}\sum_i^N \mathcal{I}(\mathbf{x}_t = \mathbf{x}_t^i)$$

where $\mathcal{I}$ is the indicator function. However, we do not actually have access to $P(\mathbf{x}_t|y_{1:t})$; indeed, that is what we are trying to estimate in the first place. Instead, we can draw samples from a known distribution q and set importance weights w_t^i as follows

$$\begin{aligned} w_t^i \propto \frac{P(\mathbf{x}_t|y_{1:t})}{q(\mathbf{x}_t|y_{1:t})} &= \frac{P(\mathbf{x}_t|\mathbf{x}_{t-1}, y_t)P(\mathbf{x}_{t-1}|y_{1:t-1})}{q(\mathbf{x}_t|\mathbf{x}_{t-1}, y_t)q(\mathbf{x}_{t-1}|y_{1:t-1})} \\ &= \frac{P(\mathbf{x}_t|\mathbf{x}_{t-1}, y_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1}, y_t)} w_{t-1}^i \\ &= \frac{P(y_t|\mathbf{x}_t)P(\mathbf{x}_t|\mathbf{x}_{t-1})}{q(\mathbf{x}_t|\mathbf{x}_{t-1}, y_t)} w_{t-1}^i \end{aligned}$$

Here, the numerator consists of two models assumed to be known: the emission model $P(y_t|\mathbf{x}_t)$ and the transition model $P(\mathbf{x}_t|\mathbf{x}_{t-1})$. Given some choice of $p(\mathbf{x}_t|\mathbf{x}_{t-1}, y_t)$ such as a simple Gaussian, we can then compute the weights recursively[5].

In summary, particle-filters perform sequential state-estimation of $P(\mathbf{x}_t|y_{1:t})$ by (1) drawing samples from $q(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i, y_t))$, (2) updating the weights using the recursion above, and (3) estimating the posterior $P(\mathbf{x}_t|y_{1:t})$ via the weighted samples.

[5] For example, one common option is to set $q(\mathbf{x}_t|\mathbf{x}_{t-1}, y_t) = P(\mathbf{x}_t|\mathbf{x}_{t-1})$ itself, which further simplifies the weights update to $w_t^i \propto P(y_t|\mathbf{x}_t)w_{t-1}^i$.

Particle Degeneracy Vanilla particle filters suffer from the well-known problem called *particle degeneracy*. Namely, after a few iterations, the particle weights begin to accumulate on a small number of particles, i.e., the weights for most particles go towards zero, while that for a few particles go towards one. As one might expect, this has a detrimental effect on how well the posterior is approximated by the samples – in the limit of only a single particle having all the weight, we are essentially trying to approximate a distribution with a single point, which is bound to not be accurate. This is captured by the notion of effective sample size (ESS) $\sum_i (w_t^i)^2$, which goes to 1 as the distribution of weights converges to a dirac-delta over a single sample. Intuitively, estimating the posterior based on N weighted samples is somewhat equivalent to estimating the same using ESS number of samples from the true posterior.

Importance Resampling To mitigate particle degeneracy, we need to draw new *resamples* from the posterior $P(\mathbf{x}_t|y_{1:t})$, and reset the weights to be uniform across all resamples. However, we only have access to the weighted samples $\{(w_t^i, \mathbf{x}_t^i)\}$ we already have. The simplest solution is to just draw resamples from this set based on the

weights. This method known as sampling-importance-resampling (SIR), suffers from a different problem: particle *impoverishment*. Specifically, as the weights accumulate on a few particles, the new set of resamples drawn proportional to weights will be dominated by these few samples. As with particle degeneracy, the reduction in particle diversity also has negative effects on the accuracy of the sample-based posterior estimation.

The OT Method Wang and Solo (2020) propose a particle filtering algorithm for Lie groups, with a particular focus on mitigating particle degeneracy and particle impoverishment through a specialized resampling algorithm. Given a set of weighted samples, the OT resampling approach proceeds in two steps. First, we draw new samples with standard sampling-importance resampling (SIR), i.e., we pick samples (with replacement) from the original set of particles according to their weights. To combat particle impoverishment, the second step of the OT approach computes a transportation map[6] from the empirical distribution of the original particles to that of the SIR resamples using the manifold's geodesic distance as the ground metric. The final output is a fresh set of samples matching the distribution of the SIR resamples but with higher particle diversity.

[6] We provide background on optimal transport theory and transportation maps in Section 2.3.2.

Concretely, the OT resampling problem consists of a set of particles $\{\mathbf{x}_i\}$, and a different set of particles $\{\hat{\mathbf{x}}_i\}$ obtained with standard SIR. The goal is to obtain a new set of particles $\{\tilde{\mathbf{x}}_i\}$ that matches the empirical distribution of the SIR resamples $\{\hat{\mathbf{x}}_i\}$. We obtain this new set $\{\tilde{\mathbf{x}}_i\}$ by applying a transportation map T to the original samples $\{\mathbf{x}_i\}$. The transportation map T defines a mapping from the distribution of the original samples to the empirical distribution of the SIR resamples, and is computed in two steps. First, we solve the a Kantorovich relaxation of the optimal transport problem between $\eta(\mathbf{x})$ and $\nu(\hat{\mathbf{x}})$. Given a cost matrix $\mathbf{C}$ of pairwise distances between samples $\{\mathbf{x}_i\}$ and $\{\hat{\mathbf{x}}_j\}$, we minimize the following entropy regularized Sinkhorn distance objective (Cuturi, 2013):

$$\gamma = \underset{\gamma \in J(\eta,\nu)}{\operatorname{argmin}} < \gamma, \mathbf{C} >_F - \epsilon H(\gamma) \quad \text{s.t} \quad \boldsymbol{\gamma}\mathbf{1} = \eta, \boldsymbol{\gamma}^T\mathbf{1} = \boldsymbol{\nu}, \tag{8.4}$$

where $\langle\rangle_F$ is the Frobenius dot-product, $J(\eta, \nu)$ is the space of all joint distributions γ with marginals η and ν, $H(\gamma)$ is the entropy of γ, and ϵ controls the level of entropy regularization[7].

[7] Recall the convention that bold-faced $\boldsymbol{\nu}$, $\boldsymbol{\eta}$, and $\boldsymbol{\gamma}$ denote vector-representations – here, of the corresponding empirical distributions.

In our experiments, we solve Equation 8.4 using the Sinkhorn-Knopp matrix scaling algorithm as implemented in the `POT`: Python Optimal Transport package (Flamary and Courty, 2017). We populate the cost matrix $\mathbf{C}$ using geodesic distances for manifolds listed in Table 8.1. For all manifolds used in our experiments, we empirically

evaluated the threshold for λ_{thresh} above which kernel matrices were observed to be SPD, and list them in Table 8.1.

Manifold	Geodesic Distance $d_{\mathcal{M}}(\mathbf{x}, \mathbf{y})$	λ_{thresh}
S^3 Hypersphere	$\arccos\left(\mathbf{x}^T\mathbf{y}\right)$	Always SPD
SO(3) Rotation matrices	$\left(\sum_{k=1}^{n} \theta_k^2\right)^{1/2}$ $\{e^{i\theta_k}\}$ are eigenvalues of $\mathbf{x}^T\mathbf{y}$	0.4
SPD matrices	$\left\|\texttt{logm}\left(\mathbf{x}^{-\frac{1}{2}}\mathbf{y}\mathbf{x}^{-\frac{1}{2}}\right)\right\|_F$	1.0

Table 8.1: **Geodesic distances for Riemannian Manifolds used in Experiments**. `logm` is the matrix log operation. The λ_{thresh} values are computed empirically for geodesic Laplacian kernels

Once the optimal joint distribution γ has been computed, the new resamples $\tilde{\mathbf{x}}_i$ are obtained by solving the following weighted Riemannian centroid problem

$$\tilde{\mathbf{x}}_i = T(\mathbf{x}_i) = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{j=1}^{N} \boldsymbol{\gamma}_{i,j} d^2(\mathbf{x}, \hat{\mathbf{x}}_j), \tag{8.5}$$

where $d(\mathbf{x}, \hat{\mathbf{x}}_j)$ is the geodesic distance between $\mathbf{x}$ and $\hat{\mathbf{x}}_j$, and elements of the joint distribution $\boldsymbol{\gamma}_{i,j}$ serve as the weights. Thus, the original samples $\mathbf{x}_i$ get transported to the weighted centroid of SIR resamples.

For resampling over Lie groups such as SO(3), Wang and Solo (2020) provide a simple iterative algorithm to compute this Riemannian centroid using the Cayley transform and the Cayley lifting map. Computing the Riemannian centroid is not a trivial problem in general, but we can obtain estimates through optimizers such as gradient descent, with the caveat that we may be stuck at local optima. In our experiments, we solve this Riemannian centroid problem in general using Riemannian gradient descent, and for experiments on SO(3), we also use the approach in Wang and Solo (2020), which we refer to as the "Cayley centroid" method.

Cayley Centroids Wang and Solo (2020) propose to compute the Riemannian weighted centroid by using the Cayley transform and the Cayley lifting map where $\texttt{Cay}_{\mathbf{x}}$ and $\texttt{Cay}_{\mathbf{x}}^{-1}$ are the Cayley transform and the Cayley lifting map at point $\mathbf{x}$. We refere the reader to Wang and Solo (2020) for additional details and references on the use of Cayley transforms, and their use to estimate the Riemannian centroid for Lie groups. The full algorithm for the OT approach as used in our experiments in provided in Algorithm 2.

Algorithm 2: Resampling using the OT Method

Data: A set of weighted samples $\{w_i, \mathbf{x}_i\}$

$\{\hat{\mathbf{x}}_i\} \leftarrow$ `SIR-Resampling`$(\{w_i, \mathbf{x}_i\})$

if *use_cayley_method* **then**

  # Cost matrix with Cayley lifting map as geodesic dist.

  $\mathbf{C}_{ij} \leftarrow \|\mathtt{Cay}_{\mathbf{x}_i}^{-1}(\hat{\mathbf{x}}_j)\|_F$

  # Solve OT problem

  $\boldsymbol{\gamma}_{i,j} \leftarrow$ `SolveSinkhonObjective`$(\mathbf{C}, \boldsymbol{\nu}, \epsilon)$

  # Solve for Centroid with Cayley Lifting & Transport Maps

  while $\|\tilde{\mathbf{x}}_j\|_F > \epsilon$ **do**

    $\mathbf{s}_i \leftarrow \frac{1}{N}\sum_{j=1}^{N} \boldsymbol{\gamma}_{i,j}\mathtt{Cay}_{\tilde{\mathbf{x}}_i}^{-1}(\hat{\mathbf{x}}_j)$

    $\tilde{\mathbf{x}}_i \leftarrow \mathtt{Cay}_{\tilde{\mathbf{x}}_i}(\mathbf{s})$

else

  # Cost matrix with geodesic dists. from Table 8.1

  $\mathbf{C}_{ij} \leftarrow d_{\mathcal{M}}(\mathbf{x}_i, \hat{\mathbf{x}}_j)$

  # Solve OT problem

  $\boldsymbol{\gamma}_{i,j} \leftarrow$ `SolveSinkhonObjective`$(\mathbf{C}, \boldsymbol{\nu}, \epsilon)$

  # Solve for Centroid with Riemannian Gradient Descent

  $\tilde{\mathbf{x}}_i \leftarrow \mathrm{argmin}_{\mathbf{x}} \sum_{j=1}^{N} \boldsymbol{\gamma}_{i,j}\mathbf{C}_{i,j}$

Algorithm 3: Resampling using Kernel Herding

Data: Set of samples $\{\mathbf{x}_i\}$ & geodesic distance function $d_{\mathcal{M}}$ for the manifold.

if *use_riemannian_herding* **then**

  # Geodesic Laplacian kernel with $d_{\mathcal{M}}$ in Table 8.1

  $k(\mathbf{x}, \mathbf{y}) = \exp(-\lambda d_{\mathcal{M}}(\mathbf{x}, \mathbf{y}))$

  `Opt = RiemannianOptimizer`

else

  # Euclidean Laplacian kernel

  $k(\mathbf{x}, \mathbf{y}) = \exp(-\lambda\|\mathbf{x} - \mathbf{y}\|_2)$

  `Opt = EuclideanOptimizer`

for $t \in [0, T]$ **do**

  $\tilde{\mathbf{x}}_t \leftarrow$ `Opt` $\left(\mathrm{argmin}_{\mathbf{x}} -\sum_{i=1}^{n} w_i k(\mathbf{x}, \mathbf{x}_i) + \mathcal{I}(t > 1)\sum_{j=1}^{t-1} \frac{1}{t} k(\mathbf{x}, \tilde{\mathbf{x}}_j)\right)$

  if *not use_riemannian_herding* **then**

    # Project with projections in Algorithm 4

    $\tilde{\mathbf{x}}_t = \mathtt{Project}_{\mathcal{M}}(\tilde{\mathbf{x}}_t)$

8.4.2 *Experiments*

We compare our approach for Riemannian kernel herding against the OT Resampling method, as well as standard Euclidean herding followed by a projection onto the relevant manifold. We test[8] all three methods on the task of resampling from an empirical distribution over two Riemannian manifolds that are commonly used to parameterize rotation matrices: the $SO(3)$ manifold of rotation matrices and the S^3 manifold of unit-quaternions.

We summarize our approach to resampling using Riemannian kernel herding in Algorithm 3. For all manifolds, we use the geodesic Laplacian kernel, defined as $k(\mathbf{x}, \mathbf{y}) = \exp\left(-\lambda\ d_{\mathcal{M}}(\mathbf{x}, \mathbf{y})\right)$, where $d_{\mathcal{M}}$ is the geodesic distance for the manifold. As these kernels tend to be SPD only above some threshold bandwidth for non-Euclidean manifolds, we empirically calculate this threshold via the protocol used in Feragen and Hauberg (2016) and Jaquier et al. (2020). Specifically, we draw 100 uniformly distributed random samples for the manifold, and compute the geodesic Laplacian kernel matrix over a range of 50 bandwidths and empirically estimate the bandwidth above which kernel matrices are SPD with probability $\sim$1. For each bandwidth, we repeat this process 10 times and calculate the proportion of the resulting kernel matrices that are positive semi-definite, i.e., have non-negative eigenvalues. When tuning kernel parameters, we only search over bandwidths above this empirical threshold. Note that this step is not required for the spherical manifold, for which the geodesic Laplacian kernel is always SPD.

Resampling on SO(3) In Figure 8.3, we compare the OT approach against Riemannian kernel herding on the task of resampling from a set of 750 SO(3) matrices with normally distributed positive weights. We created 6 such datasets, using the first one to tune model parameters, and the remaining 5 to evaluate them. For kernel herding, we performed a random hyperparameter search over kernel bandwidths and learning rates for Adam. For the OT approach, we similarly tuned its entropy regularization.

We evaluate the resamples via their average *sampling errors* on test sets, i.e., the first Wasserstein distance[9] to the weighted targets using the manifold's geodesic distance as the ground metric. As shown in Figure 8.3, we experiment with computing the Riemannian centroid required for the OT approach using the iterative Cayley maps proposed in Wang and Solo (2020) (labeled *OT + Cayley Centroid*) as well as using gradient descent (labeled *OT*). We find that Riemannian kernel herding is able to match the final sampling error from the OT approaches with about 500 fewer samples. We also present results

[8] **Experiment Infrastructure:** All our experiments were performed on a desktop with 16 Intel Core i9-9900K 3.60GHz CPUs, and 34.4 GB RAM. We used the Riemannian Adam algorithm from the Geoopt package (Kochurov et al., 2020; Bécigneul and Ganea, 2019) for optimization on all manifolds, except for the SO(3) manifold since this was not supported in Geoopt at the time of the experiments. For SO(3), we use the conjugate gradient method with adaptive line-search from the Pymanopt package (Townsend et al., 2016).

[9] See Section 2.3.2 for background on Wasserstein distances

Algorithm 4: Project$_{\mathcal{M}}$: Projections onto Manifolds

Data: Sample $\mathbf{x}$ and the manifold $\mathcal{M}$ onto which it should be projected

if $\mathcal{M} = SO(3)$ **then**
  # Project onto orthogonal matrices with determinant +1
  $s \leftarrow 1$ if $\texttt{Determinant}(\mathbf{x}) > 0$ else -1
  $\mathbf{D}, \mathbf{U} \leftarrow \texttt{SymmetricEigenDecomposition}(\mathbf{x}^T\mathbf{x})$
  # Sort descending by eigenvalues: $\lambda_1 \geq \lambda_2 \geq \lambda_3$
  $\mathbf{D}, \mathbf{U} \leftarrow \texttt{SortByEigVals}(\mathbf{D}, \mathbf{U})$
  $D = D^{-1/2} * \texttt{diag}([1.0, 1.0, s])$
  $\mathbf{x}_{proj} = \mathbf{x}\left(\mathbf{U}\mathbf{D}\mathbf{U}^T\right)$
if $\mathcal{M} = S^3$ **then**
  # Normalize
  $\mathbf{x}_{\text{proj}} \leftarrow \mathbf{x}/\|\mathbf{x}\|_2$
if $\mathcal{M} = SPD$ **then**
  # Following procedure from Higham (1988)
  $\mathbf{A} \leftarrow \frac{1}{2}(\mathbf{x} + \mathbf{x}^T)$
  $\mathbf{D}, \mathbf{U} \leftarrow \texttt{SVD}(\mathbf{A})$
  $\mathbf{H} \leftarrow \mathbf{U}\texttt{diag}(\boldsymbol{s})\mathbf{U}^T$
  $\mathbf{V} \leftarrow \frac{1}{2}(\mathbf{A} + \mathbf{H})$
  $\boldsymbol{x}_{proj} \leftarrow \frac{1}{2}(\mathbf{B} + \mathbf{B}^T)$
return $\mathbf{x}_{proj}$

for Euclidean kernel herding, where we use the Euclidean Laplacian kernel and Euclidean gradient descent (with Adam) but heuristically project the generated samples onto the SO(3) manifold. Riemannian kernel herding is able to match the final sampling error of this Euclidean approach with around 200 fewer resamples.

Resampling on S^3 Besides matrices on SO(3), rotations can also be parameterized by unit-quaternions, which are elements of the spherical manifold S^3. In fact, quaternions are often preferred over rotation matrices as they use fewer parameters, and are more computationally efficient. We repeat the earlier experiment for the S^3 manifold and report results in Figure 8.3. As before, we find that Riemannian kernel herding outperforms both the OT approach as well as Euclidean kernel herding.

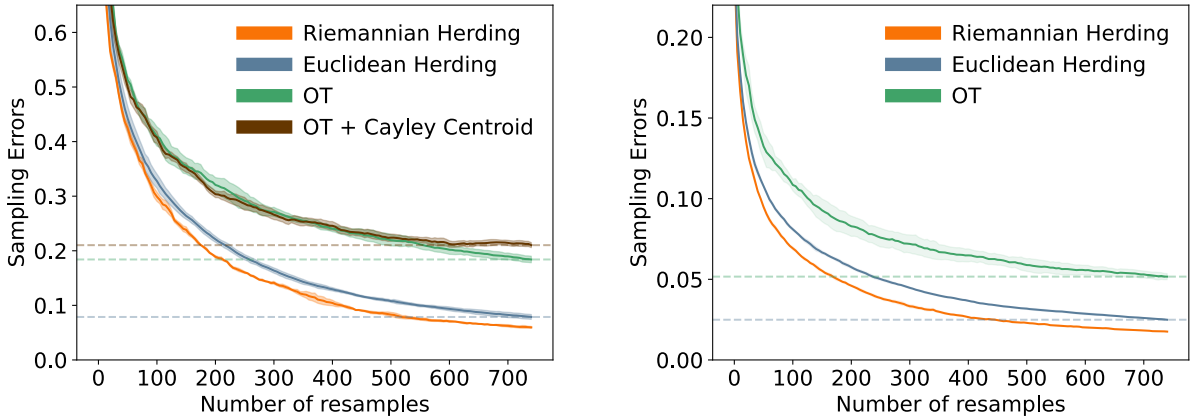


Figure 8.3: **Resampling from empirical distributions over rotations** Error bars represent standard deviation of the mean error over 5 test sets. (Left) Resampling on SO(3) (rotation matrices). (Right) Resampling on S^3 hyperspheres (quaternions). Error bars represent one standard deviation of the mean error over 5 test sets.

Comparing Algorithms The faster convergence of Riemannian kernel herding over the OT approach is likely due to negative sample auto-correlations induced by Equation 8.2 (Chen et al., 2010). This conjecture is supported by the fact that Euclidean and Riemannian herding, both of which induce negative auto-correlations, outperformed the OT methods. Furthermore, while the OT approach transports particles towards the distribution of SIR resamples, kernel herding draws samples directly from the original weighted empirical distribution.

However, the OT approach does have the advantage of being computationally faster and highly parallelizable across samples, while kernel herding tends to be slower and is inherently sequential. Thus, in practice, Riemannian kernel herding is likely to provide an advantage over the OT approach when it is worth spending additional

computation to produce high quality resamples. This may be the case, for instance, in particle filters with computationally expensive transition or observation models.

As a final comparison between the two approaches, we note that Riemannian kernel herding provides greater flexibility by allowing the weights on target samples to be negative. In traditional particle filtering problems, negative weights is generally not an issue. Indeed, if weights are meant to be discrete approximations of probabilities, negative weights are not even conceptually valid. However, negative weights routinely rise in methods operating directly with kernel mean embeddings. We now focus on one such method for simulator parameter estimation where negative weights prevent us from using the OT approach.

8.5 Simulator Parameter Estimation

In our second set of experiments, we demonstrate how adapting kernel herding to Riemannian manifolds allows us to do the same for other algorithms reliant on it. Specifically, we focus on the problem of estimating the parameters of simulators using the kernel recursive approximate Bayesian computation (KR-ABC) algorithm (Kajihara et al., 2018).

8.5.1 Kernel Recursive Approximate Bayesian Computation

Simulator Parameter Estimation Estimating the parameters of simulators is a ubiquitous problem, and is a central task in domains where real-world systems are complex, inaccessible or fragile. Simulators provide a safe and cost-effective means of experimentation in many domains including climate modeling (e.g., Palmer, 2012), epidemiology (e.g., Ferguson et al., 2020), engineering design (e.g., Bendsøe & Sigmund, 2003), and so on. In robotics, it is very common to first train policies on simulations of actual robots, choosing from a wide range of physics simulators such as MuJoCo (Todorov et al., 2012), Isaac Gym/Sim (Makoviychuk et al., 2021), Bullet (Coumans and Bai, 2016), and so on.

As a concrete example of model parameter estimation, consider the task of training a controller or a *policy* for robots, e.g., grasping an object using a robotic hand. Since training policies on real robots can be expensive, time-consuming, and prone to accidents, generally such policies are trained on a simulation of the robot, and then transferred to the real robot. The quality of this transfer can be very sensitive to how well the simulated robot matches the real one. Often robot simulations will allow users to tune its parameters, e.g. the moments

of inertia of the robot hand. But the tricky part is knowing what values to set since we may not know the corresponding values for the real robot. For instance, even if the robot's manual provides values for its moments of inertia, they may not be accurate after wear-and-tear or if we modify the robot with additional weights. Thus, there is a need to estimate these parameters from data collected on the real robot, so we can accurately calibrate the simulated robot.

For certain parameters, it may be possible to make direct measurements, e.g. mass of an object. But this may not always be possible, might require specialized equipment, or require substantial effort. A more general approach would be to adjust the parameters of the simulated robot so that for a set of fixed controls, it generates the same observations as the real robot. For instance, the moment of inertia of the palm of a robotic hand will affect how it moves in respond to fixed torque commands. We could thus tune the inertia parameters so that the simulated hand results in the same trajectory as the real hand under identical toque commands. We now discuss how kernel recursive approximate Bayesian computation allows us to do this without resorting to brute force grid search in the parameter space.

Kernel Recursive Approximate Bayesian Computation Consider a black-box[10] simulator that takes as input a parameter $\boldsymbol{\theta}$, and produces observations $\mathbf{y}$ conditioned on it. Given some target observation $\mathbf{y}^*$, our goal is to find parameters that are likely to have generated it.

[10] The simulator is "black-box" since we do not assume knowing its underlying model – we simply need to be able to run it with a specific parameter choice.

In KR-ABC, we begin by initializing a set of n parameters $\{\boldsymbol{\theta}_i\}_{i=1}^n$, and query the simulator to obtain corresponding observations $\{\boldsymbol{y}_i\}_{i=1}^n$. Given kernels k_θ and k_y for the parameters and observations respectively, we estimate the kernel mean embedding of the distribution $P_{y^*}(\boldsymbol{\theta})$ of parameters likely to have produced $\mathbf{y}^*$ as follows

$$\begin{aligned} \boldsymbol{\mu}_{P_{y^*}(\boldsymbol{\theta})} &= \sum_{i=1}^{n} w_i k_{\boldsymbol{\theta}}(\cdot, \boldsymbol{\theta}_i) \\ \text{where } [w_1, \ldots, w_n]^T &= (\mathbf{G} + n\delta\mathbb{I})^{-1}\mathbf{k}_y(\mathbf{y}^*) \end{aligned} \tag{8.6}$$

Here, $\mathbf{k}(\mathbf{y}^*) = [k_y(\mathbf{y}_1, \mathbf{y}^*), \cdots k_y(\mathbf{y}_n, \mathbf{y}^*)]^T$, $\mathbf{G} = [k_y(\mathbf{y}_i, \mathbf{y}_j)]_{i,j=1}^n$, $\mathbb{I}$ is the identity matrix and $\delta > 0$ is a regularization constant. The calculation of the (potentially negative) weights above corresponds to a kernel-ridge regression (Kajihara et al., 2018). At each iteration, kernel herding is used to draw samples from $\boldsymbol{\mu}_{P_{y^*}(\boldsymbol{\theta})}$, which are then applied to the simulator to generate observations for the next iteration. After the final iteration, we return the first herded sample as our estimated parameter.

The above KR-ABC algorithm works well when the parameter $\boldsymbol{\theta}$

lies on a Euclidean manifold. However, this is often not the case – since simulators are generally designed to match physical phenomena, their underlying models are informed by physical constraints, which are often encoded as Riemannian manifolds. In our experiments, we consider scenarios where $\boldsymbol{\theta}$ lies on a specific Riemannian manifold – the SPD[11] manifold – and evaluate the efficacy of our proposed Riemannian kernel herding as the means of drawing samples from kernel mean embeddings. We compare our approach against a few alternative baselines, which we now discuss.

[11] KR-ABC with Riemannian kernel herding is agnostic to the choice of the manifold. We picked the SPD manifold since it allowed us to examine two common scenarios – estimating covariances and inertia matrices. Moreover, since SPD matrices admit the Cholesky decomposition, it provided an alternative baseline besides heuristic projections.

Baselines: Projections and Decompositions While our proposal for Riemannian kernel herding explicitly integrates the geometry of manifolds into the sampling process, there are some alternative options that are both conceptually and computationally simpler. We use two such approaches as baselines for comparison. The first of these is the heuristic approach of simply using Euclidean kernel herding, and projecting the solution onto the required manifold using the projections defined in Algorithm 4. The second is the widely used approach of parameterizing $\boldsymbol{\theta}$ using a known decomposition for the required manifold.

As an alternative to heuristic projections, we can simply parameterize $\boldsymbol{\theta}$ in a way that ensures it always lies on the manifold. While this is not possible for all Riemannian manifolds, the SPD manifold can be conveniently parameterized using the Cholesky decomposition, i.e, any SPD matrix can be written as the product of lower-triangular matrices $\mathbf{L}$ as follows:

$$\text{Cholesky Decomposition} \quad \boldsymbol{\theta} = \mathbf{L}\mathbf{L}^{\dagger}$$

Instead of sampling the full $\boldsymbol{\theta}$ matrix directly, we can sample the elements of the lower-triangular $\mathbf{L}$ and construct the SPD $\boldsymbol{\theta}$. For $\boldsymbol{\theta}$ with $n \times n$ values, the lower triangular $\mathbf{L}$ can be parameterized with $n(n-1)/2$ values, which, importantly, do not need to be constrained.

8.5.2 Experiments

Estimating the Covariance of Gaussian Distributions In our first experiment, we set the simulator to be a 3D zero-mean Gaussian distribution, and our goal is to estimate its SPD covariance matrix using samples drawn from it. For this experiment, we created 10 datasets, each consisting of 1000 samples from the target distribution as observations; we use the first dataset to tune hyperparameters, and the rest to evaluate models.

For all methods, we set the number of herded samples $n = 100$, and tune the regularizer δ, the learning rate for Adam, and the band-

Algorithm 5: Simulator Parameter Estimation with KR-ABC

Data: Target observations $\mathbf{y}^*$; kernels k_θ and k_y; a simulator `Simulator`$(\boldsymbol{\theta})$ and a set of n random initial parameters $\{\boldsymbol{\theta}_i\}$ for it

while *not converged* **do**
  # *Simulate observations for set of parameters*
  $\{\mathbf{y}_i\} \leftarrow \{$`Simulator`$(\boldsymbol{\theta}_i)\}$
  # *Get weights from kernel ridge regression*
  $\mathbf{k}(\mathbf{y}^*) \leftarrow [k_y(\mathbf{y}_1, \mathbf{y}^*), \cdots k_y(\mathbf{y}_n, \mathbf{y}^*)]^T$
  $\mathbf{G} \leftarrow [k_y(\mathbf{y}_i, \mathbf{y}_j)]_{i,j=1}^n$
  $[w_1, \ldots, w_n]^T \leftarrow (\mathbf{G} + n\delta\mathbb{I})^{-1}\mathbf{k}_y(\mathbf{y}^*)$
  # *Build kernel mean embedding*
  $\boldsymbol{\mu}_{P_{y^*}(\boldsymbol{\theta})} \leftarrow \sum_{i=1}^n w_i k_{\boldsymbol{\theta}}(\cdot, \boldsymbol{\theta}_i)$
  # *Draw new set of parameters*
  if *use_riemannian_herding* **then**
    $\{\boldsymbol{\theta}_i\} \leftarrow$ `RiemannianKernelHerding`$(\boldsymbol{\mu}_{P_{y^*}(\boldsymbol{\theta})})$
else
  $\{\boldsymbol{\theta}_i\} \leftarrow$ `KernelHerding`$(\boldsymbol{\mu}_{P_{y^*}(\boldsymbol{\theta})}$
  $\{\boldsymbol{\theta}_i\} = \{$`Project`$_{\mathcal{M}}(\boldsymbol{\theta}_i)\})$
Return the first herded sample from last round
return $\boldsymbol{\theta}_0$

widths for k_θ and k_y (a Euclidean Laplacian kernel) through a randomized hyperparameter search. We fix the number of epochs for Adam at 100 and run KR-ABC for 20 iterations.

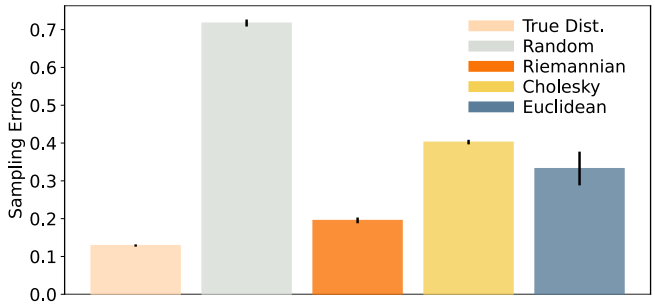


Figure 8.4: **Covariance Matrix Estimation for Gaussian Distributions**. Error bars denote one standard deviation around the mean error across 9 sets of test observations.

In Figure 8.4, we compare the different approaches in terms of their sampling errors, i.e., the first Wasserstein distance between the target samples and those generated from the estimated covariance matrices. For comparison, we also plot the sampling errors of iid samples drawn from the true distribution (labeled *True Dist*) and samples drawn from a random uniform distribution (labeled *Random*). We find that Riemannian KR-ABC outperforms all other methods, yielding sampling errors much lower than that for random uniform samples and only slightly higher than that for iid samples from the true distribution.

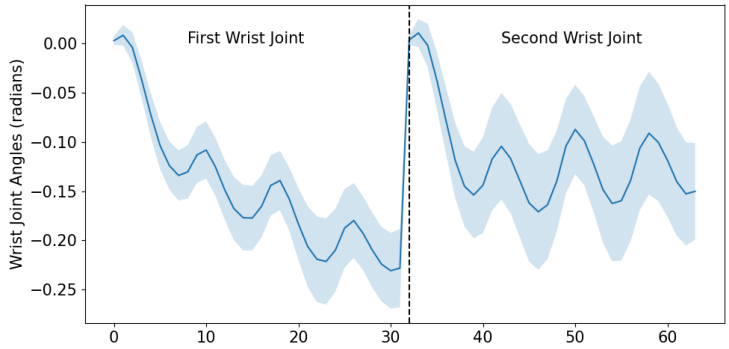


Figure 8.5: **Trajectory of Angular Positions of Two Wrist Joints of a Simulated Adroit Robot Hand**. Our observations are the concatenation of these two 32-step trajectories. The shaded region denotes the standard deviation within the different trajectories.

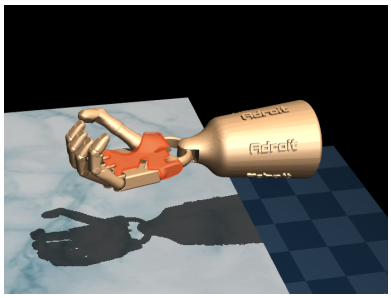


Figure 8.6: **The Adroit Robot Hand Simulator**. We estimate the inertia matrix of its palm, highlighted in red.

Estimating the Inertia Matrix of a Simulated Robot Hand Finally, we return to the motivating application for simulator parameter estimation that we laid our earlier. Namely, given a simulator of a robot-hand, we wish to tune its dynamics to match the behavior of a real robot to ensure that policies learned on the simulator transfer to the real-robot. In our experiments, we make a slight modification for experimental ease – instead of collecting data from a real robot, we collect data from a different simulator.

We apply Riemannian KR-ABC with Riemannian kernel herding to the task of estimating the 3D inertia matrix of the palm of an Adroit robotic hand simulator[12] (Figure 8.6). We use as observations the concatenated trajectories of angular positions of its two wrist joints under a fixed control sequence for 32 steps, i.e., a 64 dimensional observation (Figure 8.5). Our goal is to learn an inertia matrix that produces trajectories similar to the target observations.

We generated 6 distinct trajectories of observations by applying different sequences of controls; we use the first one to tune hyperparameters and the remaining to evaluate models. When evaluating a model's performance, we compute its *simulation error*, i.e., the average norm of the difference between the true observations and those simulated from the estimated parameters using the same controls.

[12] Environments obtained from `https://github.com/vikashplus/mj_envs` (Rajeswaran et al., 2018)

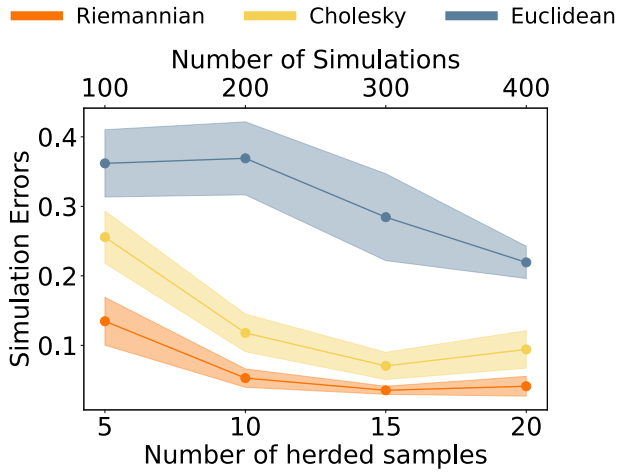


Figure 8.7: **Inertia Matrix Estimation for the Adroit Robot Hand Simulator**. Error bars denote standard error in the mean over 5 sets of observations.

In Figure 8.7, we show that Riemannian KR-ABC with Riemannian kernel herding outperforms both Euclidean KR-ABC with heuristic projections and the Cholesky parameterized variant without projections. The gains from the Riemannian approach appear to be larger for smaller number of parameter samples, which directly translates to the number of simulator queries at each iteration of the KR-ABC algorithm. Thus, Riemannian KR-ABC was able to obtain better estimates of the inertia matrix with fewer simulator queries than the other methods.

8.6 Conclusion

We presented an approach to adapting kernel herding to Riemannian manifolds by using (1) geometry-aware kernels that incorporate the correct distance metric for the manifold, and (2) Riemannian optimization to constrain generated samples to lie on the manifold. We demonstrated that our approach outperforms various alternatives on

experiments involving resampling from empirical distributions, and estimating parameters of black-box simulators. Our approach provides an intuitive and effective way of incorporating geometric information into the task of sampling from kernel mean embeddings. Possible directions for future work may be to apply a similar approach to related sampling techniques such as the Stein points algorithm (Chen et al., 2018) which also uses herding. Another direction could be to identify other algorithms that could benefit from using herding as an intermediate step. Additionally, in this paper, we chose to focus on geodesic exponential kernels due to their generality. For specific applications, it would be pertinent to explore kernels customized or learned for the task at hand, as well as kernels that are provably characteristic. Applications that benefit from incorporating kernel herding are likely to be those that require computing expectations of functions with minimal sample evaluations. One such application may be problems requiring evaluations of implicit stochastic policies in robotic control and reinforcement learning.

Part IV

Metric Informed Reinforcement Learning

9
BeigeMaps: Behavioral Eigenmaps

Abstract *Training reinforcement learning (RL) agents directly from high-dimensional observations such as images continues to be a challenging problem. Recent line of work on behavioral distances proposes to learn representations that encode behavioral similarities quantified by the bisimulation metric. By learning an isometric mapping to a lower dimensional space that preserves this metric, such methods attempt to learn representations that group together functionally similar states. However, such an isometric mapping may not exist, making the learning objective ill-defined. We propose an alternative objective that allows distortions in long-range distances, while preserving local metric structure – inducing representations that highlight natural clusters in the state space. This leads to new representations, which we term Behavioral Eigenmaps (BeigeMaps), corresponding to the eigenfunctions of similarity kernels induced by behavioral distances. We empirically demonstrate that when added as a drop-in modification, BeigeMaps improve the policy performance of prior behavioral distance based RL algorithms.*

9.1 Introduction

Reinforcement learning (RL) from image observations holds great promise as it enables learning control policies for domains where obtaining low-dimensional state information is expensive or intractable (Yarats et al., 2019; Yen-Chen et al., 2020). However, image observations are typically high-dimensional and contain redundant or task-irrelevant information. Directly applying RL on image observations often results in poor sample efficiency and weak generalization capability (Laskin et al., 2020a; Zhang et al., 2020).

Central to this challenge is the problem of representation learning (Jaderberg et al., 2016; Laskin et al., 2020b; Zhang et al., 2020; Stooke et al., 2021). The goal of representation learning is to map high-dimensional image observations to low-dimensional representations, facilitating the learning of downstream policies and/or value functions. The representation can be learned a priori (Lange and Riedmiller, 2010; Lange et al., 2012) or jointly with the policy (Yarats et al., 2019; Zhang et al., 2020; Lee et al., 2020).

Recently, a set of representation learning approaches have been proposed using behavioral distances (Zhang et al., 2020; Castro et al., 2021; Kemertas and Aumentado-Armstrong, 2021; Castro et al., 2023; Chen and Pan, 2022). These behavioral distances are variants and relaxations of the bisimulation metric (Ferns et al., 2011; Castro, 2020) and capture state-differences based on expected returns under certain policies. These representation learning approaches not only encourage the low-dimension representation to extract task-relevant information (Zhang et al., 2020), but also shape the representation space such that its metric structure aligns with task-relevant values (Castro, 2020; Castro et al., 2021). It has been shown that value function learning can benefit from the metric structure of the representation space, as states with similar values are grouped together in the representation space (Castro et al., 2021, 2023).

Although prior approaches have proposed different behavioral distances, with considerations for computation complexity (Castro et al., 2021), training stability (Kemertas and Aumentado-Armstrong, 2021), approximation error (Chen and Pan, 2022), etc., they follow the same recipe when encoding the behavior metric structure into representation space. Specifically, these methods learn a representation mapping such that the low-dimensional Euclidean representation space is *isometric*, i.e., *globally* distance preserving, to the space defined by corresponding behavioral distances.

Unfortunately, such an objective can be ill-defined – there may not exist a low-dimensional Euclidean space such that distance in the Euclidean space is equal to the behavioral distance. This issue may undermine the quality of the learned representation and induce instability in training since the behavior distances are computed through bootstrapping (Van Hasselt et al., 2018).

When exact isometry to a low-dimensional Euclidean space is not possible, we can still obtain approximate embeddings, but these necessarily come at the cost of metric distortions. This raises a natural question: if isometry is ill-defined without distortions, is there a certain *type* of acceptable distortion that still preserves important geometric structure defined by the behavior distance?

We hypothesize that a good trade-off is to allow distortion in long-range distances while preserving the *local* geometric structure given by states that are closer together in terms of behavioral distance. This proposal is further motivated by the fact that locality preserving maps are known to emphasize natural clusters in data (Belkin and Niyogi, 2001). Since a key motivation for behavioral distances is value-based state aggregation (Castro et al., 2021), it may be beneficial to use locality preserving embeddings that are inherently better suited for this purpose.

We propose representations corresponding to the top[1] eigenfunctions of kernels defined with respect to behavioral distances, which we refer to as **B**ehavioral **Eigen**maps (BeigeMaps). These eigenmaps correspond to the simultaneous optimal solution for locality preservation (Belkin and Niyogi, 2001) and state space partitioning (Von Luxburg, 2007). We demonstrate that BeigeMaps can be used as a drop-in modification of existing behavioral distance algorithms, and it improves performance for all algorithms when evaluated on the Deep Mind for Control (DMC) benchmark (Tassa et al., 2018).

[1] We use top and bottom eigenfunctions to refer to those corresponding to the largest and smallest non-zero eigenvalues respectively.

Representation Learning for RL Despite early success of RL with image input (Mnih et al., 2015), learning visual policies only relying on the RL objective, i.e., return maximization, is challenging in domains with less structured visual input (Zhang et al., 2020; Lamb et al., 2022). Representation learning for RL seeks to improve sample efficiency (Laskin et al., 2020a; Yarats et al., 2021a), robustness (Zhang et al., 2020; Lamb et al., 2022), and generalization capability (Zhang et al., 2020; Laskin et al., 2020a) by learning a good, often low dimensional, representation for control.

A group of representation learning for RL approaches follow largely from self-supervised learning on visual tasks. Representations are shaped through auxiliary tasks such as reconstructing image observations (Lange and Riedmiller, 2010; Lange et al., 2012; Hafner et al., 2019a,b; Lee et al., 2020; Yarats et al., 2019), minimizing contrastive losses (Oord et al., 2018; Laskin et al., 2020b; Stooke et al., 2021; Zhang et al., 2022), clustering (Yarats et al., 2021b; Liu et al., 2023), and encouraging consistency under data augmentation (Laskin et al., 2020a; Yarats et al., 2021a). These approaches, however, may suffer when image observations contain complex distractor signals that are irrelevant to reward and/or control (Zhang et al., 2020; Fu et al., 2021; Wang et al., 2022; Lamb et al., 2022).

Fu et al. (2021) and Wang et al. (2022) propose to explicit model distractor signals and factor such signals out for control. Another idea is to learn representations through one-step (Pathak et al., 2017; Badia et al., 2020; Baker et al., 2022) or multi-step (Efroni et al., 2021; Lamb et al., 2022) action prediction. Although these approaches reduce irrelevant information, they do not explicitly induce structure in the representation space which directly helps policy learning (Zhang et al., 2020). Jaderberg et al. (2016) learn representation through many pseudo-reward functions, though the relevance of pseudo-rewards may be domain specific. Our work follows more closely the line of work on learning representations based on behavior distances (Givan et al., 2003; Ferns et al., 2011), which we summarize below.

Representation Learning based on Behavioral Distances The behavioral distance between two states describes how "behaviorally" different they are, either under arbitrary policies (Givan et al., 2003; Ferns et al., 2011) or a fixed policy (Castro, 2020). One well-established example is the bisimulation metric (Ferns et al., 2004, 2011), which is recursively defined based on both the difference between immediate reward and the distance between the distribution of next states (given by the bisimulation metrics). Zhang et al. (2020) connect the bisimulation metric with representation learning. They propose to learn a low-dimensional representation such that the L_1 distance in the representation space approximates a behavior distance similar to the bisimulation metric. The induced metric structure facilitates reinforcement learning, as the bisimulation metric provides an upper bound on the difference of values (Ferns et al., 2011; Ferns and Precup, 2014). New behavior distances have since been proposed for representation learning to simplify calculation (Castro et al., 2021), improve training stability (Kemertas and Aumentado-Armstrong, 2021), and reduce distance approximation errors (Chen and Pan, 2022). Castro et al. (2023) introduces a kernel interpretation of behavior distances which yields a new avenue for theoretical analysis. We discuss and compare the technical details of these approaches in Section 9.9.

A key limitation of these existing approaches is that the behavior distance to be modeled may not be realizable by the distance in the low-dimensional representation space (Castro et al., 2023). When combined with bootstrapping and the use of function approximators, this realizability issue may undermine the quality of the learned representation and induce instability in training. In this chapter, we propose to mitigate this challenge by focusing on preserving *local* metric structure, while allowing long-range distortion. We construct such representations through the eigenfunctions of graph Laplacians defined with respect to behavioral distances. Such locality-preserving representations are well-known in the dimensionality reduction and manifold learning literature (Belkin and Niyogi, 2001). In the context of behavioral distance based RL, we show that these representations outperform existing approaches which try to preserve global metric structure (Zhang et al., 2020; Castro et al., 2023; Kemertas and Aumentado-Armstrong, 2021; Chen and Pan, 2022).

Laplacian Eigenmaps for Reinforcement Learning Given the ubiquity of spectral decompositions and their intimate relationship to dimensionality reduction, it is not surprising that spectral methods have been used widely in reinforcement learning. Much of this line of work can be traced back to proto value functions (PVFs) (Mahade-

van and Maggioni, 2007), reward-agnostic basis functions derived from the eigenspectrum of the Laplacian encoding the transition dynamics of a Markov decision process (MDP). Subsequent works have expanded these ideas, including using Krylov subspace methods (Petrik, 2007; Ghosh and Bellemare, 2020) and singular value decompositions (Duan et al., 2019). Ghosh and Bellemare (2020) provide an analysis of many such algorithms in terms of their accuracy, stability, and ease of estimation.

Other notable extensions include incorporating actions in such representations that traditionally only encode state information (Ren et al., 2022), and extensions to continuous state spaces Wu et al. (2018). Moreover, while spectral representations traditionally only encoded the metric structure of transition dynamics, various works have found advantages in incorporating rewards as well. Our proposed BeigeMaps also follow in this tradition of reward-aware spectral representations, and is most closely aligned with the method from Comanici and Precup (2011), which we describe below.

Comanici and Precup (2011) propose to incorporate rewards into PVFs by defining graph Laplacians with respect to bisimulation metrics. Our proposed BeigeMaps also build on the same premise, and can even be viewed as extending the work in Comanici and Precup (2011) along three axes. Firstly, Comanici and Precup (2011) study the use of Laplacian eigenmaps for policy evaluation, i.e., estimating the value of a known policy. Moreover, the bisimulation metric is also either assumed to be known, or is computed a priori. As we discuss in Section 9.4, the high computational cost of estimating this metric makes it infeasible to compute it to convergence – the computation is likely better spent simply learning the policy directly. We propose estimating Laplacian eigenfunctions within an end-to-end learning algorithm where both the metric and the policy are learned simultaneously. Secondly, Comanici and Precup (2011) assume a tabular setting, which allows for exact matrix decompositions that scale poorly with the size of the state space. We use neural Laplacian eigenmaps to learn representations for continuous state spaces, and swap expensive matrix computations for neural approximations (Deng et al., 2022a). Finally, while Comanici and Precup (2011) study the bisimulation metric, we use the on-policy variant that resolves inherent pessimism in the former (Castro, 2020).

Notation We formalize the reinforcement learning problem setting as a Markov decision process (MDP) defined by the tuple $(\mathcal{X}, \mathcal{A}, r, \mathcal{P}, \gamma)$. Here, $\mathcal{X}$ and $\mathcal{A}$ are continuous state and action spaces with dimensions $\texttt{dim}(\mathcal{X})$ and $\texttt{dim}(\mathcal{A})$, $r : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. The distribution $\mathcal{P}^a_{\mathbf{x}} = \mathcal{P}(\mathbf{x}'|\mathbf{x}, a)$

encodes transition probabilities from $\mathbf{x}$ to $\mathbf{x}'$ under action a.

The RL problem corresponds to training a policy $\pi : \mathcal{X} \to \mathcal{A}$ that maximizes the expected discounted return $\mathbb{E}_\pi\big[\sum_0^H \gamma^t r^a_{\mathbf{x}_t}\big]$, where $r^a_{\mathbf{x}} = r(\mathbf{x}, a)$, and H is the (potentially infinite) time horizon. The expectation $\mathbb{E}_\pi$ is taken over $\mathcal{P}^\pi_{\mathbf{x}}$, the transition distribution under the policy π. A policy's *value* $V^\pi(\mathbf{x}) = \mathbb{E}_\pi\big[\sum_{t=0}^H \gamma^t r^a_{\mathbf{x}_t} | \mathbf{x}_0 = \mathbf{x}\big]$ is the expected return of following π when starting at $\mathbf{x}$. We use the notation $\mathcal{P}^a_{\mathbf{x}} = \mathcal{P}^a(\mathbf{x}) = \mathcal{P}^a(\mathbf{x}'|\mathbf{x})$ and $r^a_{\mathbf{x}} = r^a(\mathbf{x}) = r(\mathbf{x}, a)$ interchangeably. The expected values of these quantities under a policy π are denoted as $d^\pi_{\mathbf{x}} = d^\pi(\mathbf{x}) = \mathbb{E}_{a\sim\pi}\big[r(\mathbf{x}, a)\big]$ and $\mathcal{P}^\pi_{\mathbf{x}} = \mathcal{P}^\pi(\mathbf{x}) = \mathbb{E}_{a\sim\pi}\big[\mathcal{P}^a(\mathbf{x}'|\mathbf{x})\big]$.

Even though we use the MDP formalism, the environments we use in our experiments actually have partial observability. Instead of formulating the problem explicitly as a partially observable MDP, we follow the commonly used practice of treating multiple stacked image observations as an approximation of state.

9.2 *Behavioral Distances*

Behavioral distances[2] define a notion of state dissimilarity that encodes differences in returns under a sequence of actions. They are termed "behavioral" precisely because they capture long-range temporal differences. These distances use a particular notion of dissimilarity, the bisimulation metric (Ferns et al., 2004, 2011), that forms an upper bound over state-value differences. This enables learning representations that group states by their value based similarities, which can be useful for downstream policy learning.

[2] We provide background on the bisimulation metric underlying behavioral distances in Section 2.4, including discussion on their connections to value functions.

The Bisimulation (Pseudo) Metric The bisimulation metric $d_\sim(\mathbf{x}, \mathbf{y})$ between two states $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ is small if they yield similar immediate rewards, and transition to similar states (Ferns et al., 2004, 2011). It is defined as the unique fixed point of the following recursion

$$d_\sim(\mathbf{x}, \mathbf{y}) = \max_{a\in\mathcal{A}} \big[\, |r^a_{\mathbf{x}} - r^a_{\mathbf{y}}| + \gamma \mathcal{W}_1(\mathcal{P}^a_{\mathbf{x}}, \mathcal{P}^a_{\mathbf{y}} \,|d_\sim) \,\big]. \tag{9.1}$$

Here, the 1-Wasserstein distance $\mathcal{W}_1(\cdot|d_\sim)$ *lifts* the ground distance $d_\sim$ (defined on $\mathcal{X}$) onto a distance between probability distributions (Peyré et al., 2019). Despite its name, $d_\sim$ is actually a *pseudo*-metric since non-identical states can still have zero distance (Ferns et al., 2011).

On-Policy Bisimulation Metric Having to enumerate over $\mathcal{X}$ makes $d_\sim$ intractable for continuous state spaces. Moreover, the maximization over actions makes it overly pessimistic – large distances may arise due to actions rarely executed under a policy (Castro, 2020).

The *on-policy* bisimulation metric d^π resolves these issues by swapping the maximization over actions with expectations over *policy-induced* transitions (Castro, 2020). It is the unique fixed point of the following iteration:

$$d^\pi(\mathbf{x}, \mathbf{y}) = |\mathbb{E}_\pi(r_\mathbf{x}) - \mathbb{E}_\pi(r_\mathbf{y})| + \gamma \mathcal{W}_1(\mathcal{P}^\pi_\mathbf{x}, \mathcal{P}^\pi_\mathbf{y} \mid d^\pi). \tag{9.2}$$

The distances $d_\sim$ and d^π define upper bounds on the differences in optimal values $|V^*(\mathbf{x}) - V^*(\mathbf{y})| \leq d_\sim(\mathbf{x}, \mathbf{y})$ and the given policy's values $|V^\pi(\mathbf{x}) - V^\pi(\mathbf{y})| \leq d^\pi(\mathbf{x}, \mathbf{y})$ respectively (Ferns et al., 2011; Castro, 2020). It is this connection to values that makes bisimulation metrics particularly useful for representation learning. If states are embedded in a space where their distances match d^π or $d_\sim$, states with similar value based behaviors should be grouped together.

Summarizing Key Properties of Bisimulation Metrics Our motivation in utilizing the theory of bisimulation metrics above is to induce useful representations for policy learning. Let us summarize some of the main properties[3] of $d_\sim$ discussed thus far as they pertain to representation learning in RL.

[3] See Section 2.4 for a detailed background on the properties listed here, including theorems on convergence and bounds (Ferns et al., 2004, 2011; Castro, 2020; Kemertas and Jepson, 2022)

1. *Bisimulation Operator Iterations Converge* Repeated applications of the bisimulation iteration in Equation 9.1 on an arbitrary semi-metric d converges to a fixed point $d_\sim$, the bisimulation metric as noted in Theorem 1. In practice, this gives us a constructive (albeit expensive) method of estimating the bisimulation metric from data.

2. *Aggregated MDP Values Approximate True Values* Using the bisimulation metric, we can aggregate states of an MDP by constructing an $(\epsilon, d_\sim)$-aggregated MDP, where all states with $d_\sim(\mathbf{x}, \mathbf{y}) \leq 2\epsilon$ for a small $\epsilon > 0$ are mapped onto a single aggregate state. Such an aggregation allows us to bound the gap between the optimal values of the aggregated and original MDPs. Consequently, solving the (potentially simpler) aggregated MDP can serve as a reliable approximation for solving the original MDP.

3. *Optimal Values are Lipschitz Continuous with respect to $d_\sim$* The bisimulation metric $d_\sim$ forms an upper bound on the difference in optimal values V^* for a pair of states. This connection to values makes bisimulation metric particularly attractive for representation learning and state aggregation – two states that are close together with respect to $d_\sim$, will also have similar values. Thus, state aggregation or representation learning with $d_\sim$ can reveal value-based clusters in $\mathcal{X}$.

4. *Bisimulation Metrics are Optimal Value Functions of Self-Coupled MDPs* The interpretation of bisimulation metrics as optimal value functions of self-coupled MDPs provides further insight into their close connection ton value functions.

The on-policy bisimulation metric d^π also essentially retain the same properties as that for $d_\sim$ above – just with reference to values of a particular policy π (Castro, 2020). Most importantly,

1. The π-bisimulation operator converges, and d^π is its unique fixed point.
2. Value functions are Lipschitz continuous with respect to d^π

$$|V^\pi(\mathbf{x}) - V^\pi(\mathbf{y})| \leq d^\pi(\mathbf{x}, \mathbf{y})$$

> In the sequel, 'bisimulation metric' or 'bisimulation distance' will refer to the on-policy variant d^π, unless specified otherwise.

Generalized Behavioral Distances Various modifications to the on-policy d^π have been proposed to improve computation and sample-based estimation, resulting in new behavioral distances (Zhang et al., 2020; Castro et al., 2021, 2023; Kemertas and Aumentado-Armstrong, 2021; Chen and Pan, 2022) that still retain the same form in Equation 9.2:

$$d^\pi(\mathbf{x}, \mathbf{y}) = d_R^\pi(\mathbf{x}, \mathbf{y}) + \gamma d_T^\pi(\mathcal{P}_\mathbf{x}^\pi, \mathcal{P}_\mathbf{y}^\pi \mid d^\pi). \tag{9.3}$$

Here, the first component d_R^π measures immediate reward-differences between states, while d_T^π measures differences in terms of their transition distributions.

Offering a kernelized perspective, Castro et al. (2023) swap distances in Equation 9.3 for kernel functions $k(\mathbf{x}, \mathbf{y})$ that measure *similarities* between states x and y. Kernels are often defined with respect to a distance; for instance, the Gaussian kernel $k(\mathbf{x}, \mathbf{y}) = \exp(-\eta\|\mathbf{x} - \mathbf{y}\|_2^2)$ with bandwidth $\eta > 0$ uses the L_2 distance. Kernels can also be defined without direct reference to a distance, as is the case in Castro et al. (2023). Despite the changes, the similarity measure given by Castro et al. (2023) still retains the form as Equation 9.3: $k^\pi(\mathbf{x}, \mathbf{y}) = k_R^\pi(\mathbf{x}, \mathbf{y}) + \gamma k_T(\mathcal{P}_\mathbf{x}^\pi, \mathcal{P}_\mathbf{y}^\pi \mid k^\pi)$. Since much of the literature uses the distance perspective, we follow the same and discuss the kernel interpretation in Section 9.10.

Behavioral Distances for Representation Learning While most properties of $d_\sim$ and d^π suggest their usefulness to developing

value-informed representations for RL, their interpretation as optimal values of self-coupled MDPs raises some potential questions. For instance, it would seem to imply that an algorithm that learns the bisimulation metric over a state space would need to essentially solve a larger MDP to find good representations for a smaller MDP. Would it be computationally beneficial then to simply avoid representation learning altogether and just solve the original MDP? To answer this question, we now discuss the computational complexity of computing bisimulation metrics, and how it informs the design of representation learning algorithms for RL.

9.2.1 *Computational Complexity of the Bisimulation Metric*

While bisimulation metrics have useful properties that make them attractive for representation learning in RL, their applicability hinges on how easily we can compute an estimate for the metric. To obtain some intuition, let us take a look at the complexity of computing these metrics for finite MDPs.

A single application of the bisimulation recursion requires $\mathcal{O}(|\mathcal{A}||\mathcal{X}|^5 \log |\mathcal{X}|)$ steps. For the on-policy variant, replacing the maximization over actions with expectations gives us savings of a factor of $|\mathcal{A}|$. To converge to $d_\sim$ or d^π with an accuracy of δ, we require $\frac{\log \delta}{\log c}$ steps. Thus, the overall complexity of estimating the bisimulation metric is $\mathcal{O}\left(|\mathcal{X}|^5 \log |\mathcal{X}| \frac{\log \delta}{\log c}\right)$, which we summarize in Table 9.1.

If we compare the cost of estimating bisimulation metrics with the cost of simply learning the optimal value via policy iteration, we run into a somewhat disappointing result. For a finite MDP, exact policy iteration converges in $\mathcal{O}\left(\frac{|\mathcal{X}||\mathcal{A}|}{1-\gamma} \log \frac{1}{1-\gamma}\right)$ steps (Scherrer, 2013), which for large state spaces is cheaper than computing bisimulation metrics by a factor of $|\mathcal{X}|^3$. This is not surprising given that bisimulation metrics correspond to optimal values of self-coupled MDPs with a quadratically larger state-space[4]. These complexity results lead us to two observations about using bisimulation metrics for representation learning in RL:

1. **Need for Joint Policy-Metric Training** When learning representations based on bisimulation metrics, the approach of first estimating the bisimulation metric and *then* using it to find a representation to aid downstream policy learning may not be particularly useful. We might be better off simply training the policy directly, without introducing an expensive pre-training phase to learn a representation[5]. Instead of waiting for the metric to converge, the information in early metric estimates could already be helpful in

Table 9.1: **Time Complexities of Computing the Bisimulation Metric** $N_T = \log \delta / \log c$ is the number of iterations required to estimate bisimulation metrics with accuracy δ. Here, $\mathcal{W}_p$ and $\mathcal{W}_p^\zeta$ are the p-Wasserstein distance and the parameterized Sinkhorn distance. See Kemertas and Jepson (2022) for details on bisimulation metrics with respect to generalized Sinkhorn distances.

	Complexity
Policy Iteration	$\mathcal{O}\left(\frac{\lvert\mathcal{X}\rvert\lvert\mathcal{A}\rvert}{1-\gamma} \log \frac{1}{1-\gamma}\right)$
$d_\sim$ (with $\mathcal{W}_p$)	$\lvert\mathcal{X}\rvert^5\lvert\mathcal{A}\rvert \log \lvert\mathcal{X}\rvert N_T$
d^π (with $\mathcal{W}_p$)	$\lvert\mathcal{X}\rvert^5\lvert\mathcal{A}\rvert \log \lvert\mathcal{X}\rvert N_T$
$d_\sim$ (with $\mathcal{W}_p^\zeta$)	$\lvert\mathcal{X}\rvert^4\lvert\mathcal{A}\rvert \log \lvert\mathcal{X}\rvert N_T$
d^π (with $\mathcal{W}_p^\zeta$)	$\lvert\mathcal{X}\rvert^4 \log \lvert\mathcal{X}\rvert N_T$
$\mathcal{W}_p$	$\mathcal{O}\left(\lvert\mathcal{X}\rvert^3 \log(\lvert\mathcal{X}\rvert)\right)$
$\mathcal{W}_p^\zeta$	$\mathcal{O}\left(\lvert\mathcal{X}\rvert^2 \log \lvert\mathcal{X}\rvert\right)$

[4] Since self-coupled MDPs have a state space of size $|\mathcal{X}|^2$, we should expect estimating this metric to be slower than policy iteration at least by a factor $|\mathcal{X}|$; the remaining factor of $|\mathcal{X}|^3$ comes from the cost of estimating Wasserstein distances.

[5] The complexities discussed here are for finite MDPs. With continuous MDPs, the use of function approximators change the analysis since fixed point iterations are replaced by gradient updates. But it is pertinent to use results from discrete MDPs to guide our expectations in the continuous case.

training the downstream policy. In fact, this is exactly how modern representation learning algorithms use bisimulation metrics (Zhang et al., 2020; Castro, 2020; Kemertas and Jepson, 2022), as we now discuss.

2. **Need for Alternatives to $\mathcal{W}_1$** The cost of computing bisimulation metrics is dominated by the cost of computing Wasserstein distances. Specifically, each step of the bisimulation operator incurs the worst-case cost of $\mathcal{O}\left(|\mathcal{X}|^3 \log |\mathcal{X}|\right)$ in estimating the Wasserstein distance (Orlin, 1988). Thus, strategies that speed up the Wasserstein distance computation can significantly speed up estimation of bisimulation metrics. Indeed, much of recent efforts in the literature have been dedicated to finding approximations and alternatives to the Wasserstein distance while still retaining useful properties of bisimulation metrics (Ferns and Precup, 2014; Castro, 2020; Zhang et al., 2020; Castro et al., 2021; Chen and Pan, 2022; Castro et al., 2023; Kemertas and Jepson, 2022). Recently, Kemertas and Jepson (2022) have formulated bisimulation metrics that swap $\mathcal{W}_1$ for the broader family of Sinkhorn distances, many of which can be computed faster and with greater parallelization[6]

[6] Sinkhorn distances bound Wasserstein distances, and define a spectrum of distances which can be traversed by setting appropriate hyperparameters. Sinkhorn distances can not only be computed faster ($\mathcal{O}\left(|\mathcal{X}|^2 \log |\mathcal{X}|\right)$ using the Sinkhorn-Knopp algorithm (Altschuler et al., 2017; Dvurechensky et al., 2018)) but can also be parallelized across multiple GPUs.

We now discuss approaches to tackle the two issues identified above. In Section 9.9, we discuss how various existing approaches have attempted to replace the expensive $\mathcal{W}_1$ computations with simpler alternatives, which can be encapsulated by a generalized formualtion of d^π. But first, we discuss how the metric and the policy can be trained jointly without having to wait for the former to converge.

9.3 *Approximate Policy Iteration with π-bisimulation*

An obvious concern with jointly learning the bisimulation metric with the policy is whether the training iterations actually converge, or if the intermittent changes in estimates of the metric cause policy learning to diverge. To illustrate the viability of this approach, we now describe frameworks for training RL agents using bisimulation metrics by interleaving metric and policy training. First, we discuss the formulation from Kemertas and Jepson (2022) for approximate policy iteration with π-bisimulation. This formulation uses fixed point iterations to estimate d^π and use it to define an (ϵ, d^π)-aggregated MDP[7], which is then supplied to policy improvement module. This formulation yields useful convergence results. In practice, gradient-updates are used in lieu of fixed point iterations to update d^π, which is then used to train a state representation, instead of performing hard state-aggregation.

[7] See Section 2.4 for details on $(\epsilon, d_\sim)$-aggregated MDPs, where all states with $d_\sim(\mathbf{x}, \mathbf{y}) \leq 2\epsilon$ for a small $\epsilon > 0$ are mapped onto a single aggregate state

Approximate Policy Iteration with π-bisimulation Kemertas and Jepson (2022) study properties of approximate policy iteration (API) coupled with *hard* state aggregation via on-policy bisimulation metrics. As shown in the pseudocode in Algorithm 6, the algorithm essentially consists of three steps. First, given an initial policy π, we apply its corresponding π-bisimulation operator $\mathfrak{F}^\pi$ to a an initial distance function for m iterations. While this fixed point iteration is not run to convergence, larger values of m will lead to better approximations of the true d^π for the current policy. Second, we use the current estimate of d^π to perform a *hard* state-aggregation, to obtain a (ϵ, d^π)-aggregate MDP with state space $\tilde{\mathcal{X}}$ from the current state space $\mathcal{X}$. Finally, we perform a policy evaluation step to estimate V^π for π evaluated on the aggregated state space $\tilde{\mathcal{X}}$. We then obtain an improved policy π_g via a policy improvement step. In the final step, π is updated as a linear combination of itself with π_g, with α controlling the degree to which π is allowed to change at every iteration.

Algorithm 6: Approximate Policy Iteration with π-bisimulation (Kemertas and Jepson, 2022)

Data: $c = \gamma$, initial distance function d^π, initial policy π, hard state-aggregation tolerance ϵ, num. bisim operator iterations m

for $i = 1$ **to** $\cdots$ **do**
  # Estimate π-bisimulation metric
  for $i = 1$ **to** m **do**
    $d^\pi \leftarrow \mathfrak{F}^\pi(d^\pi)$
  # Obtain New State Space
  $\tilde{\mathcal{X}} \leftarrow \texttt{HardAggregation}(\mathcal{X}, d^\pi, \epsilon)$
  # Update Policy
  $V^\pi \leftarrow \texttt{PolicyEvaluation}(\tilde{\mathcal{X}}, \pi)$
  $\pi_g \leftarrow \texttt{PolicyImprovement}(\tilde{\mathcal{X}}, V^\pi)$
  $\pi \leftarrow (1-\alpha)\pi + \alpha\pi_g$

Value Convergence for API with π-bisimulation For Algorithm 6, Kemertas and Jepson (2022) present the following result[8] on the convergence of the estimated value V^π to the optimal value V^* for the MDP:

Theorem 15 (Kemertas and Jepson (2022))**.** ***Value Convergence of API with π-bisimulation*** *In Algorithm 6, if $m > \log\left(\frac{1-\gamma}{1+\gamma}\right) / \log(\gamma)$ and we*

[8] Kemertas and Jepson (2022) also provide convergence results for a simpler algorithm where (1) we do not warm start the π-bisimulation fixed-point iterations with the current estimate for d^π, but the initial d_0, and (2) we set $\alpha = 0$. With these assumptions, Kemertas and Jepson (2022) show that

(a) If the learned policy converges to a policy $\bar{\pi}$

$$\|V^{\bar{\pi}} - V^*\|_\infty \leq \frac{\delta}{1-\gamma} + \frac{2\gamma(2\epsilon + c_n)}{(1-\gamma)^2}$$

(b) If the learned policy does not converge, then the policy π_k at the k-th iteration satisfies

$$\lim_{k\to\infty} \sup \|V^{\pi_k} - V^*\|_\infty \leq \frac{\delta}{(1-\gamma^2)} + \frac{2\gamma(2\epsilon + c_n)}{(1-\gamma)^3}$$

Thus, while the policy may oscillate, the deviation of the value function from the optimal value function is bounded.

use $\mathcal{W}_p$ as the probability metric in the π-bisimulation operator $\mathfrak{F}^\pi$, then

$$\lim_{k\to\infty} \sup \|V^{\pi_k} - V^*\|_\infty \leq \frac{\delta}{(1-\gamma)^2} + \frac{2\gamma(2\epsilon + \bar{\alpha} c_n)}{(1-\gamma)^3}$$

where π_k is the learned policy at the k-th iteration, V^* is the optimal value of the MDP, δ is the tolerance for the policy improvement step such that $\|T_{\pi_g} V - TV\|_\infty \leq \delta$ where T_{π_g} is the Bellman operator with respect to π_g and $TV = \sup_\pi T_\pi V$, γ is the MDP's discount factor, ϵ is the threshold for the hard-aggregation step, $c_m = \gamma^m/(1-\gamma)$, $\alpha = (\bar{\alpha}(1-(1+\gamma)c_m)(1-\gamma)/2)^p$.

Controlling Policy Updates in API with π-bisimulation In Algorithm 6, we use the parameter α to control how much the current policy π can change every iteration. This parameter is important since we do not run the bisimulation metric estimation step to convergence and warm start each iteration with estimates from the prior iteration. Thus, drastic changes in the policy may cause the distance estimation step to diverge since the target distance is changing every iteration. Kemertas and Jepson (2022) show that the differences in d^π of two policies can be bounded by the total variational distance of the policies. Thus, small changes to the policy should incur small changes to distances, suggesting that the training should be stable with a small enough α. Formally, Kemertas and Jepson (2022) present the following result

Theorem 16 (Kemertas and Jepson (2022))**.** ***Changes in d^π with Changing Policies*** *Let π, π' be two policies with corresponding π-bisimulation metrics $d^\pi_\sim$ and $d^{\pi'}_\sim$, computed with $\mathcal{W}_p$ with $p \in [1,\infty)$ as the probability metric. Then*

$$\|d^\pi - d^{\pi'}_\sim\|_\infty \leq \frac{2c_R}{(1-c_T)^2}\left[\sup_{s\in\mathcal{X}} D_{TV}(\pi(s), \pi(s)')\right]^{1/p}$$

where $D_{TV}(\pi, \pi')$ is the total variation distance between π and π'.

While Kemertas and Jepson (2022) provide intuitive and important convergence results for API with π-bisimulation, representation learning algorithms used in practice differ in a few critical ways. Firstly, since the m-step iterations with $\mathfrak{F}^\pi$ requires enumerating over the entire state space, it is infeasible in practice. Similarly, the `HardAggregation` is also difficult to translate to large state space, especially continuous state spaces. In practice, algorithms generally learn a representation of the state (e.g. as a neural network encoder) such that the metric structure of the representation conforms to d^π.

Using the generalized formulation of d^π in Equation 9.3, we now describe how existing approaches implement the approximate policy

iteration in Algorithm 6 using gradient based optimization, and without hard state aggregation.

9.4 *Behavioral Representation Learning*

In representation learning, we seek a mapping $\phi : \mathcal{X} \to \Phi$ onto a feature space $\Phi \subseteq \mathbb{R}^D$, ideally with $D \ll \texttt{dim}(\mathcal{X})$. In the context of behavioral distance learning algorithms, we aim to encode the structure of the metric space $(\mathcal{X}, d^\pi)$ on to the representation space (Φ, d_ϕ), where d_ϕ is a distance function in Φ. We achieve this by ensuring that distances d_ϕ between representations match the behavioral distances d^π of their pre-images in $\mathcal{X}$. Since similarly valued states are close-together in $\mathcal{X}$ with respect to d^π, reproducing this metric structure should generate similar value-based clusters in Φ.

Learning Representations Behavioral distance learning algorithms follow a common framework that can be categorized along three axes. The first is the choice of the *representational distance function* d_ϕ, the parameterized Euclidean distance in Φ. The second axis is the choice for the reward and transition distances d^π_R an d^π_T. At every iteration, we compute target distances d^* between state pairs, using $d^\pi_R(\mathbf{x}, \mathbf{y})$ and $d^\pi_T(\mathcal{P}^\pi_\mathbf{x}, \mathcal{P}^\pi_\mathbf{y} | d_\phi)$. In this *target distance update* step, we bootstrap the current representational distance d_ϕ as the ground distance for d^π_T. Finally, the parameters of ϕ are updated in the third *embedding update* step to minimize an objective $\mathcal{L}^\phi$ measuring errors in approximating d^* with d_ϕ. In Algorithm 7, we summarize the basic framework of behavioral distance algorithms, with an additional policy update step to simultaneously train a policy π.

The three axes of behavioral distance algorithms:

1. How is d_ϕ parameterized?
2. How is the target distance d^* computed?
3. What objective $\mathcal{L}^\phi$ is used to update embeddings ϕ?

Algorithm 7: Behavioral Distance Representation Learning

Data: Replay buffer $\mathcal{D}$, learning rates α_d and α_π, neural network ϕ with weights W_ϕ

```
# Define parameterized representation distance
rep_dist(x, y) : return d_φ(x, y)
for i = 1 to ··· do
    {(x, a_y, r_y, x'), (y, a_y, r_y, y')} ~ D_Buffer
    # Target distance update
    d*[x, y] ← d^π_R(r_x, r_y) + γ d^π_T(P^π_x, P^π_y | rep_dist)
    # Embedding update via distance loss
    W_φ ← W_φ − α_d ∇_{W_φ} Σ_{x,y} L^φ(d*(x, y), d_φ(x, y))
    # Additional embedding update via policy loss
    W_φ ← W_φ − α_π ∇_{W_φ} PolicyLoss(φ)
```

Joint Distance-Policy Learning We can interpret Algorithm 7 as a generic deep RL algorithm with an additional representation update step (Lines 6-8) being interleaved between the usual policy updates. However, it is worth thinking about how this addition affects the algorithm's stability and convergence.

Since d^π depends on π, the policy updates after each iteration (Line 10) could theoretically cause bisimulation distance estimation (Line 6) to diverge. Zhang et al. (2020) show that if the policy π improves monotonically over iterations, estimates of d^π will converge to $d^{\pi^*}_\sim$ using the fixed point iterations in Equation 2.3. However, monotonic policy improvement may not be guaranteed (Kemertas and Aumentado-Armstrong, 2021), and the use of d_ϕ to boostrap estimates of introduce further complications. Moreover, as we have discussed in Section 9.3, the distance $\|d^\pi_\sim - d^{\pi'}_\sim\|_\infty$ between policies π and π' can be bounded by their total variational distance. Since small updates to the policy should then result in small changes to d^π, we can expect distance estimation in Algorithm 7 to remain stable as long as the policy does not change too much between iterations. Such conservative policy updates are already commonly used in RL literature, for instance in TRPO (Schulman et al., 2015) and PPO (Schulman et al., 2017).

In summary, the potential instabilities of jointly training d^π and π can be mitigated through conservative policy updates. Moreover, in the tabular setting, such joint training does indeed converge to the optimal value. While Algorithm 7 introduces many elements that differ from the ideal tabular setting, these changes can generally be viewed as relaxations of their tabular counterparts.

Having discussed computational aspects of interleaving representation learning in Algorithm 7, let us now take a closer look at the representation learning objective in the algorithm. As we discuss in the subsequent section, our proposal for BeigeMaps simply replaces an existing global isometry objective with an alternative locality-preserving one.

9.5 From Global Isometry to Locality Preservation

9.5.1 The IsoMap Objective

While behavioral distance algorithms have experimented with different representational distances and target update steps, they all essentially share the same embedding update step. Namely, the loss $\mathcal{L}^\phi$ corresponds to finding an isometric mapping from the distance space $(\mathcal{X}, d^\pi)$ to (Φ, d_ϕ). Drawing connections to the dimensionality reduction algorithm Isomap (Tenenbaum et al., 2000), we refer to this

as the *isomap* objective[9] defined below as an expectation over the data $\mathcal{D}$ in a replay buffer:

$$\mathcal{L}^{\phi}_{\text{IsoMap}}(d^*, d_\phi) = \mathbb{E}_{\mathbf{x},\mathbf{y}\sim\mathcal{D}}\big[\big(d^*(\mathbf{x},\mathbf{y}) - d_\phi(\mathbf{x},\mathbf{y})\big)^2\big]. \tag{9.4}$$

The isomap objective forms the basis of many dimensionality reduction algorithms (Borg and Groenen, 2005), and has been tackled using various methods including gradient based optimization as in Algorithm 7 (Kruskal, 1964). However, the objective is technically ill-defined for arbitrary distances – there may not exist a D-dimensional Euclidean space Φ where all pairwise distances d^* can be mapped exactly.

[9] This objective is often called 'stress' (Kruskal, 1964) and is the loss minimized by various isometric embedding algorithms including Isomap (Tenenbaum et al., 2000), metric multi-dimensional scaling (Borg and Groenen, 2005) , SMACOF, etc. We use the term IsoMap to stress the isometric nature of this objective.

Approximate Embeddings with Distortion Even if an exact isometric embedding does not exist, we can still find an approximate embedding if we are willing to accept some distortion in the metric. Indeed, we know from Bourgain's theorem that any metric space $(\mathcal{X}, d)$ defined on N points can be embedded in $(\mathbb{R}^{\log^2 N}, L_p)$ with distorion $\mathcal{O}(\log N)$ for every $p \geq 1$. In the context of behavioral metrics, Castro et al. (2023) tackled this question of embeddability for the reduced MICO distance, and showed that an isometric embedding is possible with the squared Euclidean distance in $\mathbb{R}^D$ with $D \leq \texttt{dim}(\mathcal{X})$. However, this upper-bound is much too loose for practical purposes where we require $D \ll \texttt{dim}(\mathcal{X})$. In this context, Castro et al. (2023) show[10] that a lower dimensional isometric embedding with $D < \texttt{dim}(\mathcal{X})$ is possible if one accepts a certain degree of distortion in the original metric.

[10] Specifically, for any $\epsilon \in (0,1)$, there exists an approximate embedding $\phi : \mathcal{X} \to \mathbb{R}^m$ such that $(1-\epsilon)d^{\pi}{}_{ksme} \leq \|\phi(\mathbf{x}) - \phi(\mathbf{y})\|_2^2 \leq (1+\epsilon)d^{\pi}{}_{ksme}$, where $d^{\pi}{}_{ksme}$ is the specific distance proposed in Castro et al. (2023). The required dimension of Φ, m, scales as $\mathcal{O}(\epsilon^2)$ with the distortion ϵ

Since some degree of metric distortion is necessary to find low-dimensional representations via Equation 9.4, we pose the following natural question:

> If isometry is ill-defined without some distortion, is there a certain *type* of acceptable distortion that still preserves important metric structure of the state space?

We hypothesize that a good trade-off is to allow distortion in long-range distances while preserving the *local* metric structure of $(\mathcal{X}, d^{\pi})$. We now present our proposal to use Laplacian eigenmaps as locality-preserving representations, and argue why they can be promising representations for RL.

9.6 *Laplacian Eigenmaps for Locality Preservation*

We motivate Laplacian eigenmaps from the perspective of locality-preservation on manifolds. As in manifold learning, we assume

that even though the state space is nominally high-dimensional, we can approximate it with a lower dimensional manifold $\mathcal{M}$. In the case of behavioral distances, we assume that $\mathcal{M}$ is one where the bisimulation distances d^π define geodesic distances. Building upon the intuitions and results from Belkin and Niyogi (2003), we now establish Laplacian eigenmaps as an appropriate locality-preserving non-liner dimensionality reduction algorithm.

A Bisimulation Manifold Let $\mathcal{M}$ be a manifold such that its geodesic distance is approximated by the bisimulation metric d^π. While we do not have access to points on the manifold $\mathbf{x}_\mathcal{M} \in \mathcal{M}$, we have observations $\mathbf{x} \in \mathcal{X}$ obtained from some observation function $\psi(\mathbf{x}_\mathcal{M}) = \mathbf{x}$. Moreover, we also have access to a distance function d^π for pairs of points in $\mathcal{X}$ that approximates the geodesic distance between their pre-images on $\mathcal{M}$:

$$\text{dist}_\mathcal{M}(\mathbf{x}_\mathcal{M}, \mathbf{y}_\mathcal{M}) \approx d^\pi(\mathbf{x}, \mathbf{y}) \quad \text{where} \quad \mathbf{x} = \psi(\mathbf{x}_\mathcal{M}) \text{ and } \mathbf{y} = \psi(\mathbf{y}_\mathcal{M}) \tag{9.5}$$

We define the notion of a local neighborhood around points $\mathbf{x}_\mathcal{M} \in \mathcal{M}$ with respect to a threshold ϵ as follows

$$B_\mathcal{M}(\mathbf{x}_\mathcal{M}) = \{\mathbf{y}_\mathcal{M} \in \mathcal{M} \mid \text{dist}_\mathcal{M}(\mathbf{x}_\mathcal{M}, \mathbf{y}_\mathcal{M}) \approx d^\pi(\mathbf{x}, \mathbf{y}) \leq \epsilon\}$$

Let $k_\mathbf{x}(\cdot) = k(\mathbf{x}_\mathcal{M}, \cdot) : \mathcal{M} \to \mathbb{R}_{\geq 0}$ correspond to a non-negative measure associated with $B_\mathcal{M}(\mathbf{x}_\mathcal{M})$. For instance, a simple option would be to set $k_x(\mathbf{y}_\mathcal{M}) = 1/\texttt{vol}(B_\mathcal{M}(\mathbf{x}_\mathcal{M}))$ if $\mathbf{y}_\mathcal{M} \in B_\mathcal{M}(\mathbf{x}_\mathcal{M})$, and 0 otherwise.

Our desiderata from a locality-preserving representation $\phi_\mathcal{M} : \mathcal{M} \to \Phi$ is that a state $\mathbf{x}_\mathcal{M}$ should have a similar representation to other states in its neighborhood $B_\mathcal{M}(\mathbf{x}_\mathcal{M})$. However, since we only have access to observations $\mathbf{x} = \psi(\mathbf{x}_\mathcal{M})$, we are actually interested in a function ϕ that acts on observed states $\mathbf{x} \in \mathcal{X}$ such that $\phi_\mathcal{M} = \phi \circ \psi$. In other wrods, the observation function ψ is implicitly defined – it has already been applied to the dataset – and we need to find ϕ that act on observed states. For notational simplicity, we will overload the function ϕ to refer to both $\phi_\mathcal{M}$ and ϕ depending on its argument, i.e,

$$\phi(\mathbf{x}_\mathcal{M}) = \phi_\mathcal{M}(\mathbf{x}_\mathcal{M}) = (\phi \circ \psi)(\mathbf{x}_\mathcal{M}) = \phi(\mathbf{x})$$

One Dimensional Locality Preserving Representations Let us first formulate the notion of a one dimensional locality preserving map $\mathcal{M} \to \mathbb{R}$. Essentially, for any point $\mathbf{s}_\mathcal{M} \in \mathcal{M}$, we would like $\phi(\mathbf{s}_\mathcal{M})$ to be mapped close to all its neighbors in $B_\mathcal{M}(\mathbf{s}_\mathcal{M})$, i.e., $|\phi(\mathbf{s}_\mathcal{M}) - \phi(\mathbf{y}_\mathcal{M})|$ should be small for all $\mathbf{y}_\mathcal{M} \in B_\mathcal{M}(\mathbf{s}_\mathcal{M})$. In other words, the

gradient of ϕ at $\mathbf{x}_{\mathcal{M}}$, $[\nabla \phi(\mathbf{x}_{\mathcal{M}})]_{\mathbf{x}_{\mathcal{M}}=\mathbf{s}_{\mathcal{M}}}$ should be minimized i.e.,

$$\phi^* = \operatorname*{argmin}_{\phi \in \mathcal{F}} \int_{\mathcal{M}} \| \left[\nabla \phi(\mathbf{x}_{\mathcal{M}}) \right]_{\mathbf{x}_{\mathcal{M}}=\mathbf{s}_{\mathcal{M}}} \| dk_{\mathbf{s}_{\mathcal{M}}}(\mathbf{x}_{\mathcal{M}}),$$

where $\mathcal{F}$ defines an appropriate hypothesis class for ϕ that excludes trivial solutions. For instance, setting ϕ to be constant over all points trivially minimizes the objective, but results in a non-informative and useless solution. The integral of the gradient operator above can be expressed in terms of the Laplace-Beltrami operator $L_{\mathcal{M}}$ as follows if we set the kernel to be the heat kernel k^H:

$$\int_{\mathcal{M}} \| \nabla \phi \|^2 = \int_{\mathcal{M}} L_{\mathcal{M}} \phi = \int_{\mathcal{M}} \left[\phi(\mathbf{x}_{\mathcal{M}}) - \phi(\mathbf{s}_{\mathcal{M}}) \right] k^H_{\mathbf{s}_{\mathcal{M}}} \tag{9.6}$$

In the second equality above, we have used the fact the Laplace-Beltrami operator measures the average difference between $\phi(\mathbf{s}_{\mathcal{M}})$ and its neighbors $\phi(\mathbf{x}_{\mathcal{M}})$ for all $\mathbf{x}_{\mathcal{M}} \in B_{\mathcal{M}}(\mathbf{s}_{\mathcal{M}})$ with respect to the heat kernel k^H. Note that the heat kernel k^H is the solution to the heat-equation on $\mathcal{M}$ defined with respect to $L_{\mathcal{M}}$.

In Equation 9.6, information about the manifold $\mathcal{M}$ is provided exclusively by the kernel k^H – specifically information about the local neighborhoods in $\mathcal{M}$. In our setting, this information is encoded by the bisimulation distance d^π, which approximates the geodesic distance $\text{dist}_{\mathcal{M}}$ on $\mathcal{M}$. Ideally, we would like to encode this distance information into k^H. It turns out that k^H can, in fact, be expressed as a Gaussian kernel with respect to the geodesic distance $\text{dist}_{\mathcal{M}}$ from Varadhan's formula as follows:

$$k^H(\mathbf{x}_{\mathcal{M}}, \mathbf{y}_{\mathcal{M}}) = \lim_{t \to 0} \exp\left(-\frac{1}{4t} \text{dist}_{\mathcal{M}}(\mathbf{x}_{\mathcal{M}}, \mathbf{y}_{\mathcal{M}})^2 \right) \approx \lim_{t \to 0} \exp\left(-\frac{1}{4t} d^\pi(\mathbf{x}, \mathbf{y})^2 \right).$$

where we have replaced $\text{dist}_{\mathcal{M}} \approx d^\pi$ as per Equation 9.5.

Putting everything together, we obtain the following locality preservation objective

$$\begin{aligned} \phi^* &= \operatorname*{argmin}_{\phi \in \mathcal{F}} \int_{\mathcal{M}} \left[\phi(\mathbf{x}_{\mathcal{M}}) - \phi(\mathbf{s}_{\mathcal{M}}) \right] \exp\left(-\eta \text{dist}_{\mathcal{M}}(\mathbf{s}_{\mathcal{M}}, \mathbf{x}_{\mathcal{M}})^2 \right) \\ &= \operatorname*{argmin}_{\phi \in \mathcal{F}} \int_{\mathcal{M}} L_{\mathcal{M}} \phi \end{aligned}$$

Equivalently, we can express this objective in terms of the states $\mathbf{x}, \mathbf{s} \in \mathcal{X}$ and the bisimulation distance as follows

$$\phi^* = \operatorname*{argmin}_{\phi \in \mathcal{F}} \int_{\mathcal{X}} \left[\phi(\mathbf{x}) - \phi(\mathbf{s}) \right] \exp\left(-\eta d^\pi(\mathbf{s}, \mathbf{x})^2 \right) \approx \operatorname*{argmin}_{\phi \in \mathcal{F}} \int_{\mathcal{M}} L_{\mathcal{M}} \phi \tag{9.7}$$

Here, the optimal solution ϕ^* essentially tries to push all points in $\mathcal{X}$ close together, but the degree to which a pair of points should be

pushed together is weighted by the heat kernel. Since the heat kernel encodes similarities with respect to $d^\pi \approx \text{dist}_\mathcal{M}$, this objective thus ensures that points that are close together with respect to d^π on $\mathcal{X}$ (or equivalently $\text{dist}_\mathcal{M}$ on $\mathcal{M}$), will be pushed close together in $\Phi = \mathbb{R}$ with respect to the Euclidean distance.

While the first equality in Equation 9.7 is more intuitive, the second equality gives us a recipe to solve the optimization problem. Namely, the solutions correspond to the eigenfunctions of $L_\mathcal{M}$.

$$[L_\mathcal{M}\phi](\mathbf{x}_\mathcal{M}) = \lambda\,[L_\mathcal{M}\phi](\mathbf{x}_\mathcal{M})$$

Since eigenfunctions are orthonormal, we have the following function class for our objective:

$$\mathcal{F} = \left(\phi : \|\phi\|_2 = 1 \text{ and } \phi \perp \phi'\ \forall \phi' \in \mathcal{F}\right)$$

Finally, we require some way to exclude the trivial constant function solution to our optimization problem. To do so, we note that a constant function would have zero gradient everywhere so

$$\text{For } \phi \in \mathcal{F} \quad \int_\mathcal{M} \|\nabla \phi\|^2 = 0 \implies \int_\mathcal{M} L_\mathcal{M}\phi = \int_\mathcal{M} \lambda\phi = 0 \implies \lambda = 0$$

Thus, excluding the trivial constant function simply amounts to excluding eigenfunctions with zero eigenvalues. The number of zero eigenvalues is determined by the connectivity of $\mathcal{M}$ – for a fully connected manifold, we have $0 = \lambda_0 < \lambda_1 < \lambda_2 \cdots$. Thus, the optimal locality preserving function ϕ^* is the second eigenfunction of $L_\mathcal{M}$.

Higher Dimensional Representations To obtain a D-dimensional representation such that the representation space $\Phi = \mathbb{R}^D$, we can solve D successive optimization problems to obtain the D entries of $\boldsymbol{\phi}$. As discussed, the first entry of $\boldsymbol{\phi}$ should be the second eigenfunction of $L_\mathcal{M}$. Now, to compute the second entry of $\boldsymbol{\phi}$, we simply require that the solution be different from the first one, i.e., it should be orthogonal to the second eigenfunction of $L_\mathcal{M}$. Thus, finding the optimal D-dimensional locality-preserving representation $\boldsymbol{\phi}$ corresponds to finding the D bottom eigenfunctions of $L_\mathcal{M}$ with non-zero eigenvalues.

Sample Based Approximation: From Manifolds to Graphs When solving the objective in practice, we generally do not have access to the true Laplace-Beltrami operator $L_\mathcal{M}$, but instead must approximate it using samples from $\mathcal{M}$. Given a dataset of N samples $\mathcal{D} = \{\mathbf{x}_\mathcal{M}\}_{i=1}^N$ drawn uniformly[11] from $\mathcal{M}$, the following *graph Laplacian* $\mathbf{L}$ is the sample-estimate of $L_\mathcal{M}$

$$\mathbf{L} = \mathbf{D} - \tilde{\mathbf{K}} \quad \text{where} \quad \tilde{\mathbf{K}}[i,j] = k^H(\mathbf{x}_{\mathcal{M}i}, \mathbf{x}_{\mathcal{M}j}) \quad \text{and} \quad \mathbf{D}[i,j] = \sum_j \tilde{\mathbf{K}}[i,j]$$

[11] We assume a uniform distribution here for simplicity, but we could also draw samples from an arbitrary distribution over $\mathcal{M}$. Belkin and Niyogi (2005) provide convergence rates for this case as well

where $\tilde{\mathbf{K}} \in \mathbb{R}^{N \times N}$ be the matrix of kernel evaluations defined as

$$\tilde{\mathbf{K}}[i,j] = k^H(\mathbf{x}_{\mathcal{M}i}, \mathbf{x}_{\mathcal{M}j}) = \exp(-\eta \text{dist}_{\mathcal{M}}(\mathbf{x}_{\mathcal{M}}, \mathbf{y}_{\mathcal{M}}) \approx \exp(-\eta d^{\pi}(\mathbf{x}, \mathbf{y}))$$

Finally, note that the sample-estimate graph Laplacian $\mathbf{L}$ converges to the true $L_{\mathcal{M}}$ as $N \to \infty$, but we obtain a faster convergence rate for the following *normalized* graph Laplacian, which is generally the operator used in practical algorithms,

$$\mathbf{K} = \mathbf{D}^{-1}\tilde{\mathbf{K}} \text{ and } \mathbf{L} = \mathbb{I} - \mathbf{K}$$

In the sequel, we will often use 'Laplacian' to refer to the normalized Laplacian, either using the normalization described above or an alternative normalization based on context.

The Laplacian Eigenmap Objective Using the above sample estimates, the Laplacian eigenmap objective can be expressed as follows

$$\phi^d_{\text{LapEig}} = \underset{\phi}{\text{argmin}} \sum_{i,j}^{N} \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|_2^2 \, \tilde{\mathbf{K}}(\mathbf{x}_i, \mathbf{x}_j) \quad \text{s. t. } \Phi^T \mathbb{D} \Phi = \mathbb{I}, \tag{9.8}$$

where $\Phi \in \mathbb{R}^{N \times D}$ is the solution matrix, the i-th row of which corresponds to the D-dimensional embedding of the i-th data point x_i. The constraint $\Phi^T \mathbf{D} \Phi$ encodes the orthonormality requirement of the eigenmaps. Under these constraints, the columns of the optimal Φ matrix correspond to the bottom D eigenvectors of the normalized graph Laplacian $\mathbf{L} = \mathbb{I} - \mathbf{K}$.

Note that the spectrum of the normalized kernel matrix $\mathbf{K}$ and that of $\mathbf{L}$ are related. Specifically,

$$\mathbf{L}\phi = \lambda\phi \implies \mathbf{D}^{-1}\tilde{\mathbf{K}}\phi = \mathbf{K}\phi = (1-\lambda)\phi$$

Thus the optimal solutions to Equation 9.8 can also be found by taking the *top* D eigenvectors of $\mathbf{K}$. This can be the better approach for eigenvector estimation algorithms that are designed to find the top eigenvectors as opposed to the bottom. This is the case for the NeuralEF algorithm that we use, and now describe in the following section.

9.6.1 Neuralizing Eigen Decompositions

While matrix decompositions enable a closed form solution for ϕ_{LapEig}, these methods scale poorly with the number of samples N

and feature map dimension D, and do not allow out-of-sample generalization to unseen states. While we assumed a fixed set of N data points in Equation 9.8, in practice, we need to generalize the representation to arbitrary states not covered in the dataset. We now describe the NeuralEF algorithm (Deng et al., 2022b), which approximates kernel eigenmaps through neural networks, enabling out-of-sample generalization without expensive matrix decompositions.

The NeuralEF Algorithm Moving beyond matrix decompositions, we can generalize the notion of eigenvectors of kernel matrices to eigenfunctions of kernel functions. Given a kernel function $k(\mathbf{x}, \mathbf{y})$, its eigenfunctions ϕ are characterized by the integral eigenvalue equation $\int_{\mathcal{X}} k(\mathbf{x}, \mathbf{y})\phi(\mathbf{y})p(\mathbf{y})d\mathbf{y} = \lambda\phi(\mathbf{x})$ for some probability measure $p(\mathbf{x})$, where $\lambda \geq 0$ are eigenvalues. The Neural Eigenfunctions algorithm (NeuralEF) approximates these eigenfunctions ϕ by minimizing the following objective derived from the kernel function's eigendecomposition (Deng et al., 2022b):

$$\phi^d_{\text{NEF}} = \underset{\phi}{\operatorname{argmin}} \underbrace{\frac{1}{B}\left(\beta \sum_{j=1}^{d}\sum_{i=1}^{j-1} \hat{\mathbf{C}}[i,j]^2 - \sum_{j=1}^{d} \mathbf{C}[j,j]\right)}_{\mathcal{L}^{\phi,d}_{\text{NEF}}}, \tag{9.9}$$

$$\text{s. t. } \mathbb{E}[\phi \odot \phi] = \vec{\mathbb{1}}\ ,\ \mathbf{C} = \Phi\hat{\mathbf{K}}^d\Phi \text{ and } \hat{\mathbf{C}} = \hat{\Phi}\hat{\mathbf{K}}^d\hat{\Phi},$$

where β is a positive coefficient, $\odot$ is the Hadamard product (element wise-multiplication), $\vec{\mathbb{1}}$ is the all-ones vector, and $\hat{\mathbf{A}}$ corresponds to a matrix $\mathbf{A}$ computed with stop-gradients. The batch kernel matrix $\hat{\mathbf{K}} \in \mathbb{R}^{B\times B}$ consists of pair-wise evaluations of $k(\mathbf{x}, \mathbf{y})$ over a batch of B states. The orthogonality constraint in Equation 9.8 has been slacked as a penalty in the first term scaled by β. To enforce the normalization constraint $\mathbb{E}[\phi \odot \phi] = \vec{\mathbb{1}}$, Deng et al. (2022b) apply L_2 batch normalization after the last layer of ϕ. While the NeuralEF algorithm does assume a positive definite kernel, it can also recover the top eigenfunctions of indefinite kernels as long as it has at least $D - 1$ positive eigenfunctions (Deng et al., 2022a).

Alternative Eigenfunction Solvers NeuralEFs are not the only approach to learning eigenfunctions capable of out-of-sample generalization. Random Fourier Features (RFFs) (Rahimi and Recht, 2007) and the Nystrom method (Nyström, 1930; Williams and Seeger, 2000) are some classic approaches. RFFs are not ideal for non shift-invariant kernels and generally require feature maps with large dimensionality D. The Nystrom method relies on explicit matrix eigendecomposition, which can be prohibitively expensive for large N.

More recent approaches include Spectral Inference Networks (SpIN) (Pfau et al., 2018) and SpectralNets (Shaham et al., 2018) . Both of these methods rely on Cholesky decompositions, which is computationally expensive and tends to introduce instabilities. We opted for NeuralEFs since they do not suffer from the drawbacks described here, as well as the fact the NeuralEFs are shown to be applicable to indefinite kernels as well (as long as the kernl has more than $D-1$ positive eigenvalues). Theoretically, BeigeMaps could be learned with any of these or other eigenfunction learning approaches.

9.7 *Locality-Preservation over Global Isometry*

Having established Laplacian eigenmaps as an appropriate representation for locality-preservation, we now discuss why this approach is particularly suitable for representation learning through behavioral metrics.

Behavioral Distances are *Upper* Bounds on Value Differences Since d^π forms an *upper* bound on value differences, local distance information is much more informative than long-range distances. Specifically, if two states $x, y \in \mathcal{X}$ have $d^\pi(\mathbf{x}, \mathbf{y}) = \epsilon$, we know that for small values for ϵ, the corresponding values must also be similar, i.e. $|V^\pi(\mathbf{x}) - V^\pi(\mathbf{y})| \leq \epsilon$. If $\epsilon = 0$, they could even be aggregated into the same representation without loss of value-relevant information. However, for large ϵ, we do not obtain much information about the states' relative values. They could have very different values or even identical values, and the bound would still be satisfied. Thus, preserving the structure of local state neighborhoods may be more important in describing the valued-based cluster structure in $\mathcal{X}$, as we now further describe.

Locality Preservation Reveals Cluster Structure The central motivation behind representations based on bisimulation metrics is to organize states into value-based clusters, which may be beneficial for downstream value or policy learning. Locality-preserving representations, specifically the Laplacian eigenfunctions we propose to use in the next section, are well-suited for revealing natural clusters in data (Belkin and Niyogi, 2001). In fact, Laplacian eigenfunctions are a fundamental component of spectral clustering algorithms since they correspond to the optimal solutions to a relaxed Ncut problem of partitioning a graph into balanced clusters (Von Luxburg, 2007). The central idea behind these algorithms is to use encode data using these locality-preserving embeddings so that even a simple clustering algorithm (e.g. kmeans) can detect clusters in the embedding space.

Thus, prioritizing locality preservation at the cost of preserving long-range distances may be beneficial in revealing value-based cluster information in the state space.

Preventing Representation Explosions and Collapses We defined the Laplacian eigenmap objective by relaxing the strict isometric objective. However, the orthonormality condition in Equation 9.8 introduces new constraints as well. So one may ask if these constraints introduce unwanted biases. Interestingly, they actually *resolve* known issues identified in existing behavioral distance algorithms.

Specifically, Kemertas and Aumentado-Armstrong (2021) find that some existing behavioral distance algorithms are prone to feature map *explosion*, where the norm of the embeddings grows beyond a required bound, and feature map *collapse*, where all embeddings collapse onto a single point in Φ. With Laplacian eigenmaps, the normalization constraint ensures that $\mathbb{E}(\|\phi\|_2^2) = D$, which prevents feature map explosion. On the other hand, the orthogonality constraint ensures that $\texttt{dim}(\Phi) = D$, preventing collapse to a sub-space with dimension less than D. These are well-known properties of Laplacian eigenmaps, and are verified in Appendix 14.1. Thus, the new constraints introduced by Laplacian eigenmaps are actually beneficial for behavioral distance based representation learning.

The above properties motivate us to seek representations that preserve local structure in behavioral distance learning algorithms. We now formulate our proposed locality preserving objective, which simply corresponds to Laplacian eigenmaps with respect to behavioral distances, i.e., Behavioral Eigenmaps or BeigeMaps.

9.8 BeigeMaps: Behavioral Eigenmaps

We propose Behavioral Eigenmaps (BeigeMaps) as an alternative behavioral distance based representation for RL. This representation simply corresponds swapping out the $\mathcal{L}_{\text{IsoMap}}$ (Equation 9.4) with that in Equation 9.9, the NeuralEF objective corresponding to the top eigenfunctions of the heat-kernel based on behavioral distances.

We now describe BeigeMaps in terms of the three axes[12] defined in Section 9.2.

Representational Distance We use the L_2 distance in Φ as the representational distance for BeigeMaps, i.e., $d_\phi(\mathbf{x}, \mathbf{y}) = \|\phi(\mathbf{x}) - \phi(\mathbf{y})\|_2$. This is because Equation 9.8 encodes (the inverse of) kernel similarities $k^d(\mathbf{x}, \mathbf{y})$ into the L_2 norm in Φ – states close together in $\mathcal{X}$ with respect to d are mapped close in Φ with respect to the L_2 distance. We thus use the L_2 distance on Φ as a proxy for the bisimulation

[12] Recall that the three axes of behavioral distance algorithms are:

1. How is d_ϕ parameterized?
2. How is the target distance d^* computed?
3. What objective $\mathcal{L}^\phi$ is used to update embeddings ϕ?

metric, as is the norm in behavioral distance algorithms.

Target Distance Update When computing target distances d^*, we simply kernelize a base behavioral distance. Specifically, given a batch of state pairs $\{(\mathbf{x}, \mathbf{y})\}_i^B$ and their corresponding target distances $d^*(\mathbf{x}, \mathbf{y})$, we compute the batch kernel matrix $\mathbf{K}^{d^*}(\mathbf{x}, \mathbf{y}) = \exp(-\eta d^*(\mathbf{x}, \mathbf{y})^2)$, where $\eta \geq 0$ is the kernel's bandwidth. As discussed in Section 9.6, this choice of the kernel corresponds to an approximation of the heat kernel.

Instead of using the kernel directly, we normalize it via the batch degree matrix $\mathbf{D} = \sum_{ij} \mathbf{K}_{ij}$ and supply it as the target for the NeuralEF objective in Equation 9.9. There are multiple choices for how the kernel matrix is normalized in the dimensionality reduction and spectral clustering literature. We consider two of the most common variants of the normalized kernel: $\mathbf{K}_{sym} = \mathbf{D}^{-1/2}\mathbf{K}\mathbf{D}^{-1/2}$ (symmetric) and $\mathbf{K}_{rw} = \mathbf{D}^{-1}\mathbf{K}$ (random-walk), following the nomenclature in Von Luxburg (2007). In our experiments (Appendix 14.5), models with $\mathbf{K}_{sym}$ outperformed those with K_{rw}, so we report results using $\mathbf{K}_{sym}$.

Embedding Update Finally, we compute the eigenfunctions of the new target kernel $k^{d^*}(\mathbf{x}, \mathbf{y})$ using the NeuralEF algorithm (Equation 9.9). The resulting embeddings are approximations of the Laplacian eigenmaps (Equation 9.8) with respect to d^*, i.e., $\phi_{\text{BeigeMap}} = \phi^{d^*}_{\text{NEF}} \approx \phi^{d^*}_{\text{LapEig}}$ and $\mathcal{L}^{\phi}_{\text{BeigeMap}} = \mathcal{L}^{\phi, d^*}_{\text{NEF}}$.

The overall objective for BeigeMaps can thus be written as

$$\phi^{d^*}_{\text{BeigeMap}} = \underset{\phi}{\operatorname{argmin}} \frac{1}{B} \left(\beta \sum_{j=1}^{d} \sum_{i=1}^{j-1} \hat{\mathbf{C}}[i,j]^2 - \sum_{j=1}^{d} \mathbf{C}[j,j] \right)$$
$$\text{s.t. } \mathbb{E}[\phi \odot \phi] = \vec{\mathbb{1}}\,, \ \mathbf{C} = \Phi \hat{\mathbf{K}}^{d^*} \Phi \text{ and } \hat{\mathbf{C}} = \hat{\Phi} \hat{\mathbf{K}}^{d^*} \hat{\Phi}, \tag{9.10}$$

where d^* is the target behavioral distance computed at every iteration in Algorithm 7 and $\mathbf{K}^{d^*}$ is the normalized Gaussian kernel matrix with respect to the distance d^*.

We have proposed BeigeMaps as a drop-in replacement for the isometric embedding objective in behavioral distance based RL algorithms. As such, BeigeMaps complement these algorithms by providing an alternate means of learning representations from existing behavioral distances. We now discuss how BeigeMaps can be incorporated into various behavioral distances proposed in the literature.

9.9 *BeigeMap Algorithms*

Behavioral distance based RL algorithms jointly learn a policy π along with a representation ϕ induced by learned distances d^{π}. As shown in Algorithm 7, training repeatedly iterates through 1) representation updates with respect to the behavioral distances, and 2) policy updates from the underlying policy learning algorithm, as well as 3) updates to any additional components such as a learned dynamics model or a reward prediction model. While behavioral distance learning is agnostic to the underlying policy learning method, much of prior work uses the Soft Actor Critic (SAC) algorithm (Haarnoja et al., 2018). In Figure 9.1, we illustrate their general framework, where blue (dark) boxes correspond to vanilla SAC components, and dashed boxes indicate various auxilliary components. The distance based representation learning components for BeigeMap algorithms are shown in yellow (light). The corresponding illustration for the baseline algorithms without BeigeMaps would simply remove the $K^{d^*}_{sym}$ component, and replace $\mathcal{L}^{\phi}_{\text{BeigeMap}}$ with $\mathcal{L}^{\phi}_{\text{IsoMap}}$.

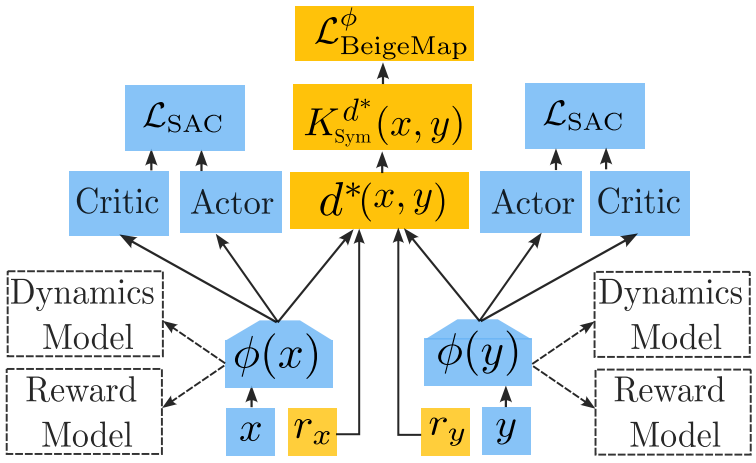


Figure 9.1: **Schematic of Behavioral Distances Based SAC Algorithms** The blue boxes are components of the underlying SAC policy learning model, dashed boxes are auxilliary components, and yellow boxes are BeigeMap components.

We now discuss behavioral distance algorithms in the literature, and their BeigeMap variants. Table 9.2 summarizes the representational distance d_{ϕ} and target distance updates for all algorithms. We also present normalized returns averaged (interquartile median) across 7 Deep Mind control environments in Figure 9.2 for reference. Experimental details and further analysis are provided in the next section.

DBC The Deep Bisimulation for Control (DBC) algorithm was the first gradient-based method to jointly learn a policy with a behavioral distance based representation for non-deterministic MDPs (Zhang et al., 2020). To compute the distance targets d^*, DBC sets

the reward-based distance d_R^π to the absolute difference in rewards, and the transition distance d_T^π to the 2-Wasserstein distance $\mathcal{W}_2$ as an approximation of $\mathcal{W}_1$ in Equation 2.3. While bisimulation metrics were originally formulated with $\mathcal{W}_1$, Kemertas and Jepson (2022) have shown that these metrics can be derived with alternative probability metrics in the family of Sinkhorn distances (Peyré et al., 2019), which includes $\mathcal{W}_2$.

Algorithm	Distance Update $d^* = d_R^\pi + \gamma d_T^\pi$	Representational Distance $d_\phi(\mathbf{x}, \mathbf{y})$	Loss: $\mathcal{L}^\phi$
DBC	$\lvert r_\mathbf{x} - r_\mathbf{y}\rvert + \gamma \left(\lVert \mu_{\phi(\mathbf{x})} - \mu_{\phi(\mathbf{y})} \rVert_2^2 + \lVert \Sigma_{\phi(\mathbf{x})}^{1/2} - \Sigma_{\phi(\mathbf{y})}^{1/2} \rVert_F^2 \right)$	$\lVert \phi(\mathbf{x}) - \phi(\mathbf{y}) \rVert_1^2$	$\mathcal{L}^\phi_{\text{IsoMap}}$
RobustDBC	$\lvert r_\mathbf{x} - r_\mathbf{y}\rvert + \gamma \left(\lVert \mu_{\phi(\mathbf{x})} - \mu_{\phi(\mathbf{y})} \rVert_2^2 + \lVert \Sigma_{\phi(\mathbf{x})}^{1/2} - \Sigma_{\phi(\mathbf{y})}^{1/2} \rVert_F^2 \right)$	$\lVert \phi(\mathbf{x}) - \phi(\mathbf{y}) \rVert_2^2$	$\mathcal{L}^\phi_{\text{IsoMap}}$
MICO	$\lvert r_\mathbf{x} - r_\mathbf{y}\rvert + \gamma d_\phi(x', y')$	$\frac{1}{2}\lVert \phi(\mathbf{x}) \rVert_2^2 + \frac{1}{2}\lVert \phi(\mathbf{y}) \rVert_2^2 + \beta\theta(\phi(\mathbf{x}), \phi(\mathbf{y}))$	$\mathcal{L}^\phi_{\text{IsoMap}}$
RAP	$\sqrt{\lvert r_\mathbf{x} - r_\mathbf{y}\rvert^2 - \sigma_r^2(x) - \sigma_r^2(y)} + \gamma d_\phi(\mu_{\phi(\mathbf{x})}, \mu_{\phi(\mathbf{y})})$	$\frac{1}{2}\lVert \phi(\mathbf{x}) \rVert_2^2 + \frac{1}{2}\lVert \phi(\mathbf{y}) \rVert_2^2 + \beta\theta(\phi(\mathbf{x}), \phi(\mathbf{y}))$	$\mathcal{L}^\phi_{\text{IsoMap}}$
X+BeigeMap	Same as algorithm X	$\lVert \phi(\mathbf{x}) - \phi(\mathbf{y}) \rVert_2^2$	$\mathcal{L}^\phi_{\text{BeigeMap}}$

Algorithm	Kernel Update $k^* = k_R^\pi + k_T^\pi$	Representational Kernel $k_\phi(\mathbf{x}, \mathbf{y})$	Loss: $\mathcal{L}^\phi$
KSME	$1 - \frac{r_\mathbf{x} - r_\mathbf{y}}{r_{\max} - r_{\min}} + \gamma k_\phi(\mathbf{x}, \mathbf{y})$	$\phi(\mathbf{x})^T \phi(\mathbf{y})$	$\mathbb{E}(k^*(\mathbf{x}, \mathbf{y}) - k_\phi(\mathbf{x}, \mathbf{y}))$
KSME+Beigemap	Same as KSME	$\exp\left(-\eta \lVert \phi(\mathbf{x}) - \phi(\mathbf{y}) \rVert_2^2\right)$	$\mathcal{L}^\phi_{\text{BeigeMap}}$

Table 9.2: **Behavioral Distances** The three components of behavioral distance (top) and similarity (bottom) based algorithms.

To further simplify $\mathcal{W}_2$ computations, DBC trains an approximate dynamics model parameterized as a Gaussian $\mathcal{N}(\mu_{\phi(\mathbf{x})}, \Sigma_{\phi(\mathbf{x})})$, with mean $\mu_{\phi(\mathbf{x})}$ and covariance $\Sigma_{\phi(\mathbf{x})}$, as well as a reward model predicting rewards from the next latent states. With this approximate Gaussian dynamics model, the $\mathcal{W}_2$ distances are calculated in closed-form as the L_2 distance between the means μ_ϕ and the Frobenius norm distance between the square-root of covariances $\Sigma_{\phi(\mathbf{x})}^{1/2}$ as shown in Table 9.2. While simplifying computation, this approximation introduces a mismatch (Kemertas and Aumentado-Armstrong, 2021). As shown in Figure 9.2, DBC+BeigeMaps significantly outperforms DBC in terms of returns averaged over all environments.

R-DBC The Robust DBC (R-DBC) (Kemertas and Aumentado-Armstrong, 2021) algorithm corrects some training instabilities that can arise with DBC. Specifically, DBC training can diverge due to embedding explosion and embedding collapse. Embedding explosions are characterized by embedding norms growing beyond the threshold[13] $\frac{r_{\max} - r_{\min}}{2(1-\gamma)}$, a stability bound derived by the authors. Embedding collapse, on the other hand, can occur for environments with sparse or near-constant rewards. In such environments, the representation ϕ can prematurely collapse onto a single point $\phi(\mathbf{x}) = \phi_0 \; \forall \mathbf{x} \in \mathcal{X}$ since the model initially observes all states to essentially have zero behavioral distance (Kemertas and Aumentado-Armstrong, 2021).

While retaining the basic behavioral distance in DBC (but fixing distances to L_2), R-DBC proposes a few algorithmic changes. To pre-

[13] Kemertas and Aumentado-Armstrong (2021) use a generalized version of the bisimulation metric leading to the bound $\frac{c_R(r_{\max} - r_{\min})}{2(1-c_T)}$ for $c_T \in [0, 1), c_R \in [0, \infty)$. For d^π in Equation 2.3, this corresponds to setting $c_T = \gamma, c_R = 1$.

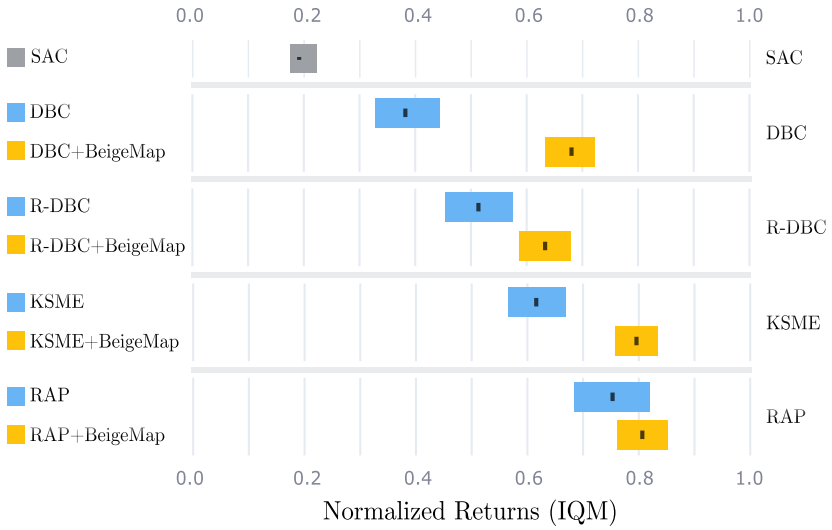


Figure 9.2: **Comparing Average Normalized Returns on DM Control Environments** Interquartile median of normalized returns of all models averaged over 30 random seeds over the 7 DMC environments. Shaded regions denote 95%-CI.

vent embedding explosion, R-DBC clips the embedding norms so they remain under the required threshold. To prevent embedding collapse, R-DBC learn an additional inverse dynamics model in the latent space predicting actions from latent state transitions, and adds a curiosity-driven intrinsic reward to the original rewards. This intrinsic reward is set proportional to errors between the predicted next latent state from the learned dynamics model and the actual next latent state, encouraging exploration to states where dynamics modeling needs improvement.

In Figure 9.2, we see that the proposed changes do indeed improve performance over DBC. But we note that both R-DBC+BeigeMaps and, interestingly, DBC+BeigeMaps outperform R-DBC. This is likely because of embedding explosion and collapse, the two issues that R-DBC solves, are both inherently mitigated by BeigeMap's orthonormality constraints as discussed in Section 9.7. Specifically, the normalization constraint encourages $\|\phi\|_2^2 \sim D$, and the orthogonality penalty encourages $\texttt{dim}(\Phi) \sim D$. In Figure 9.3, we see that $\|\phi\|$ converges to $\sqrt{D}$ for R-DBC+BeigeMap, while it continues to grow for R-DBC.

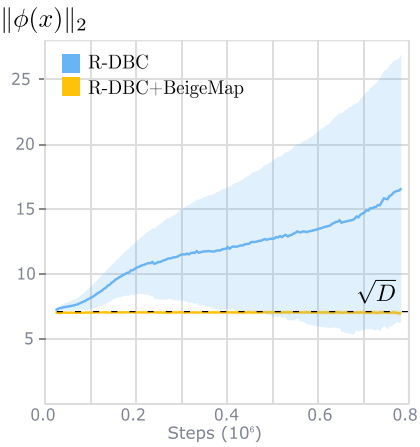


Figure 9.3: **BeiegeMaps Prevent Feature Map Explosions** Mean norm (and shaded standard deviation) of embeddings during training, averaged over all environments. The dashed line indicates $\sqrt{D} = \sqrt{50} \approx 7.1$.

KSME Castro et al. (2023) take a kernelized approach to behavioral distances, defining state *similarilaties* through kernel functions as opposed to differences. However, KSME still follows the same framework we have been using with certain kernel based modifications. Instead of representational distances d_ϕ, KSME defines representational similarities encoded as the inner-product $k_\phi(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^T \phi(\mathbf{y})$. The target distance update step is then replaced by a target kernel update

step $k^*(\mathbf{x}, \mathbf{y}) = k_R^\pi(\mathbf{x}, \mathbf{y}) + \gamma k_T(\mathcal{P}_\mathbf{x}^\pi, \mathcal{P}_\mathbf{y}^\pi \mid k_\phi)$. Corresponding to the reward differences d_R^π in other methods, KSME uses the similarity kernel $k_R^\pi(\mathbf{x}, \mathbf{y}) = 1 - \frac{r_\mathbf{x} - r_\mathbf{y}}{r_{\max} - r_{\min}}$. The transition-based similarities are encoded in the *lifted* kernel $k_T(\mathcal{P}_\mathbf{x}^\pi, \mathcal{P}_\mathbf{y}^\pi | k_\phi) = \mathbb{E}(k_\phi(\mathbf{x}, \mathbf{y}))$, where the expectation is taken over $\mathbf{x} \sim \mathcal{P}_\mathbf{x}^\pi$ and $\mathbf{y} \sim \mathcal{P}_\mathbf{y}^\pi$. Finally, KSME optimizes the kernel equivalent of the isomap objective $\mathbb{E}(k^*(\mathbf{x}, \mathbf{y}) - k_\phi(\mathbf{x}, \mathbf{y}))$, where we try to match all pair wise similarities.

Castro et al. (2023) demonstrate that KSME can be interpreted to induce a behavioral distance that is equivalent to the MICO (Matching under Independent Couplings) distance, which approximates the $\mathcal{W}_2$ under independent couplings of distributions (Castro et al., 2021). Moreover, the policies trained using KSME were shown to be empirically equivalent to those trained using MICO. Thus, we use KSME as a substitute for MICO in our experiments.

To define the BeigeMap variant of KSME, we set the representational kernel $k_\phi(\mathbf{x}, \mathbf{y})$ to be the Gaussian kernel[14] as shown in Table 9.2. We also replace the kernel matching objective with the NeuralEF objective defined over the normalized kernel obtained from KSME's target kernel updates.

[14] Recall that our choice of using the geodesic Gaussian kernel is motivated by the heat kernel. While alternate kernels, including the linear KSME kernel could also be used, we stick to the Gaussian kernel to maintain consistency across all models.

RAP The Reducing Approximation Gap (RAP) distance attempts to mitigate various approximation errors in existing behavioral distance algorithms (Chen and Pan, 2022). Firstly, RAP replaces the $\mathcal{W}_2$ distance in DBC with a sample based divergence $\mathbb{E}_{a_\mathbf{x} \sim \pi, a_\mathbf{y} \sim \pi}(d_\phi(\mathbb{E}(\mathbf{x}'), \mathbb{E}(\mathbf{y}')))$, where $\mathbb{E}(\mathbf{x}') = \mathbb{E}_{\mathbf{x}' \sim P_\mathbf{x}^{a_\mathbf{x}}}$ is an expectation over next states. This is motivated by the fact that the theoretical guarantees for the bisimulation metric derived with respect to $\mathcal{W}_1$ may not hold for $\mathcal{W}_2$. To estimate this new distributional distance, RAP trains an approximate Gaussian dynamics model with mean $\mu_{\phi(\mathbf{x})}$, and sets $\mathbb{E}(\mathbf{x}') = \mu_{\phi(\mathbf{x})}$. The representational distance d_ϕ is $\frac{1}{2}\|\phi(\mathbf{x})\|_2^2 + \frac{1}{2}\|\phi(\mathbf{y})\|_2^2 + \beta\theta(\phi(\mathbf{x})$, where $\theta(\cdot)$ is the angle between vectors.

Secondly, RAP approximates the reward-based distance when computing the target d^* as $d_R^\pi(\mathbf{x}, \mathbf{y}) = |\mathbb{E}_{a_\mathbf{x} \sim \pi} r_x^{a_\mathbf{x}} - \mathbb{E}_{a_\mathbf{y} \sim \pi} r_y^{a_\mathbf{y}}|$, whereas existing algorithms simply use $|r_\mathbf{x}^{a_\mathbf{x}} - r_\mathbf{y}^{a_\mathbf{y}}|$. This is estimated through the covariances σ_r of a learned Gaussian reward model model $\mathcal{N}(\mu_r, \sigma_r)$ as shown in Table 9.2. Finally, RAP uses a slightly modified objective $\mathcal{L}_{\text{RAP}} = \mathbb{E}_\mathcal{D}[\left(d_\phi - \gamma d_S^\pi(\mathbf{x}, \mathbf{y})\right)^2 - d_S^{\pi 2}]$ to avoid square roots. Despite this algebriac change, the new objective still corresponds to finding an isometric mapping ϕ.

9.10 Experiments

To evaluate the effectiveness of our proposal, we augment the algorithms in Section 9.9 with BeigeMaps and demonstrate improvements

in policy learning over all baseline behavioral distance algorithms.

DM Control Environment We train and evaluate all algorithms on the Deep Mind for Control (DMC) suite (Tassa et al., 2018), a set of continuous control tasks that has been a benchmark for all prior behavioral distance algorithms. We picked the following 7 environments from the suite to maximize overlap with prior work: `finger_spin`, `walker_stand`, `walker_walk`, `cheetah_run`, `cartpole_swingup`, `cartpole_balance`, and `ball_in_cup_catch`. We use RGB pixel observations of size $(3, 88, 88)$, and stack 3 consecutive frames to account for partial observability. All environments are truncated at 1000 steps and have dense rewards bounded between $[0, 1]$, except for `ball_in_cup_catch` which has sparse binary rewards. We normalize the default episodic returns within $[0, 1000]$ to $[0, 1]$ to standardize all results. Further details on environment parameters are provided in the Appendix 14.2.

Model Architectures We use the Soft Actor Critic (SAC) algorithm as the underlying RL model for all algorithms (Figure 9.1). All algorithms also learn a Gaussian latent dynamics model, parameterized by an MLP. DBC, R-DBC, and RAP algorithms also learn a latent reward prediction model parameterized by an MLP. We fixed all hyperparameters to match those used by Zhang et al. (2020), which has been a common baseline for all other prior works. Due to computational constraints, we reduced the replay buffer size from $1M$ to $100,000$. For BeigeMap algorithms, we set the kernel bandwidth using the median-heuristic (Garreau et al., 2017). We list all model hyperparameters and network architecture details in Appendix 14.3.

9.10.1 Results

We compare the performance of all baseline and BeigeMap algorithms using multiple robust evaluation metrics recommended by Agarwal et al. (2021b). For each algorithm, we used the model parameters that resulted in the highest return during training, and obtained the normalized returns from the corresponding policies for 30 random seeds. For comparison, we also present results for the vanilla SAC algorithm. In all plots, the shaded error bars correspond to 95% confidence intervals obtained through stratified bootstraping over $10,000$ samples with replacement computed using the `rliable` library (Agarwal et al., 2021b). While we report on aggregate metrics in the main text, we also provide individual learning curves in Appendix 14.4.

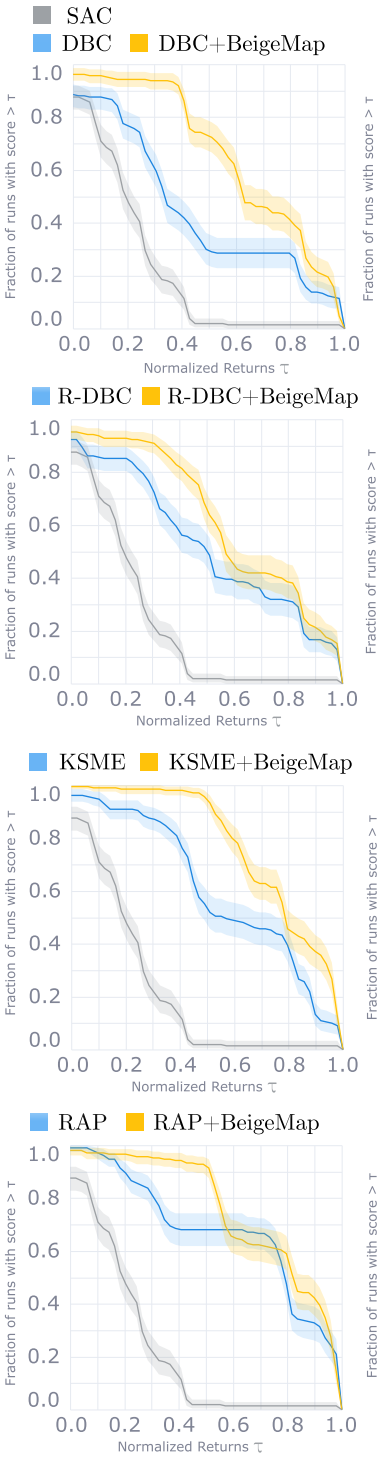


Figure 9.4: **Performance Profiles for Experiments on the Deep Mind Control Suite** Comparing algorithms with respect to multiple performance thresholds.

Interquartile Medians Figure 9.2 shows the interquartile median (IQM) normalized returns for all models averaged over all environments and evaluation trials. The BeigeMap variants (yellow) considerably outperform their respective baselines (blue) for DBC, R-DBC, and KSME. The RAP+BeigeMap variant does obtain higher average returns than RAP, but with some confidence-intervals overlap. Finally, the variations between the different BeigeMap models is lower than that between the baselines. The worst performing BeigeMap model (R-DBC + BeigeMap) still outperforms an average baseline model.

Performance Profiles Point estimates such as the IQM may not be best suited to reveal performance variability across tasks (Agarwal et al., 2021b). Thus, in Figure 9.4, we present performance profiles that tend to better capture these differences. For each model, we calculate the proportion of trials (across all environments) where the model's returns exceeded various thresholds τ. A model is said to *stochastically dominate* another if its curve is strictly above the latter (Agarwal et al., 2021b). BeigeMap variants clearly stochastically dominate their counter parts for DBC, R-DBC, and KSME. Moreover, the RAP+BeigeMap curve is strictly above RAP except for a small range of thresholds $\tau \in (0.5, 0.7)$. This provides additional context to the IQM comparisons earlier, suggesting that BeigeMap+RAP is the strictly better option for most performance thresholds.

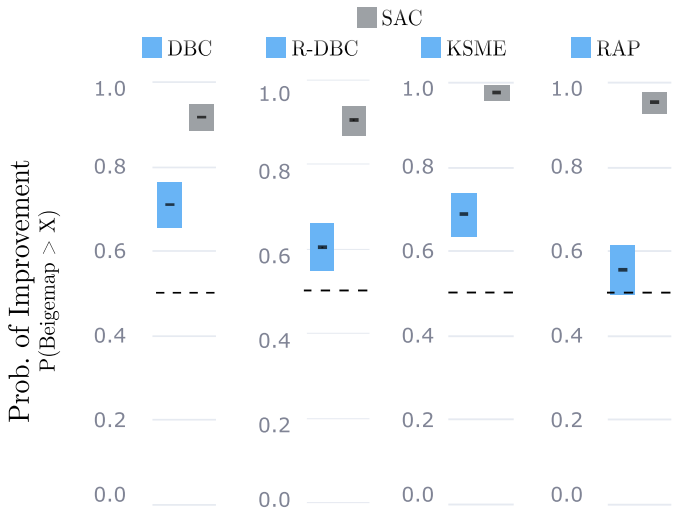


Figure 9.5: **Probabilities of Improvement for Experiments on the Deep Mind Control Suite** Likelihoods of BeigeMap algorithms outperforming their baselines.

Probability of Improvement In Figure 9.5, we plot the average probability of BeigeMap algorithms improving upon their baseline counterparts. Here, the probabilities of improvements correspond to the fraction of trials where one model obtained higher returns than

the other. Thus, this measure does not consider the magnitude of differences, but illustrates the robustness of improvement from one algorithm over another (Agarwal et al., 2021b). Improvements are considered significant if their confidence intervals do not contain 0.5. As shown in Figure 9.5, all models consistently lie above the 0.5 threshold, with the only exception of RAP+BeigeMaps, whose confidence interval crosses the 0.5 threshold by a small margin of 0.01. Overall, these results indicate that the addition of BeigeMaps is likely to improve the performance of current distance based representation learning.

9.11 Conclusion

We outlined our proposal for Behavioral Eigenmaps (BeigeMaps) for behavioral distance based representation learning in RL. While existing approaches train representations with respect to a potentially ill-defined isometry objective, we proposed an alternative locality-preserving objective. The resulting representations correspond to Laplacian eigenmaps with respect to behavioral distances, which we propose to adapt to RL problems with continuous state spaces via neural eigenfunctions. Our experiments showed that using BeigeMaps as a drop-in component in existing behavioral distance learning algorithms yieled better performing policies.

We identify two limitations of our current formulation of BeigeMaps, and discuss plans for future research directed at resolving them.

Controlling the Level of Distortion We motivated locality-preservation over the IsoMap objective by noting that the latter may be ill-defined without admitting some metric distortion. We then proposed Laplacian eigenmaps as a way to up-weight the penalty of distortions in local neighborhoods, while allowing greater distortion in long-range distances. While our analysis and experiments do show advantages of this approach, it is unclear whether Laplacian eigenmaps result in lower overall distortion than optimizing the IsoMap objective.

Practically, the level of induced distortion can be controlled by increasing the size D of the representation space. While the IsoMap objective does not provide a principled way to estimate an appropriate value for D, BeigeMaps offer some possibilities. As in most spectral decomposition methods, the eigenvalues can often inform this decision, or at least provide a measure of estimating the marginal benefit of increasing dimensionality. In future work, we plan on identifying means of quantifying the level of distortion induced by BeigeMaps, and seeking methods of controlling it through the learned eigenvalues.

Distractor Invariance Prior works have demonstrated benefits of behavioral distance based representations in terms of robustness to task-irrelevant distractors in observations (Zhang et al., 2020; Kemertas and Aumentado-Armstrong, 2021; Chen and Pan, 2022). Thus far, we have focused on demonstrating the effectiveness of BeigeMaps in RL from images in general.

Potential directions of future work on BeigeMaps include analyzing the effectiveness of BeigeMaps in inducing distractoor invariance for better generalization. Once again, a promising approach in this direction may also come from the learned eigenvalues of the Laplacian. The spectrum of kernels can be used to design efficient regularization operators that bias solutions towards functions that exhibit minimal variance within clusters (Smola and Kondor, 2003). Specifically, a regularization operator that penalizes weights associated with larger Laplacian eigenvalues could be used to design a principled approach for generalization and distractor invariance.

10
Modular Policy Composition with Policy Centroids

Abstract *We consider the task of aggregating policies in multi-objective decision making problems where multiple policies are trained to accomplish potentially conflicting tasks. While policy composition is a crucial component of multi-objective problems, commonly used techniques are restricted to simple averages that do not adhere to or exploit the structure of policy spaces. We present a new framework for policy composition viewing the problem as computing centroids in distance spaces where policies are embedded. These policy centroids not only subsume various existing composition techniques, but also provide a new means of inducing useful properties in composite policies through judicious choices of embedding spaces and distances. Additionally, we introduce policy centroids that extend existing compositions to new problem settings. For deterministic policies, we use distances between state-action value functions to define utility centroids that can be tuned to adjust the intensity of individual policy preferences. For stochastic policies, we introduce policy centroids based on statistical distances including the maximum mean discrepancy, which are particularly useful when policies are accessible only through samples. We evaluate our proposed policy centroids on various illustrative examples that highlight their benefits over existing approaches.*

10.1 Introduction

As we continue to expand the scale and complexity of sequential decision problems tackled by automated agents, it becomes increasingly difficult to quantify all desired objectives into a single monolithic goal. We are routinely faced with tasks with multiple competing objectives but no concrete notion of the composite goal. Autonomous vehicles must quickly reach their destinations while avoiding crashes; economic policy should bolster growth but also minimize inequality; financial investment strategies need to maximize returns while maintaining a diverse portfolio.

Even when we *can* define a composite objective, it may still be difficult to learn a policy to accomplish it. Indeed, one of the central dogmas of reinforcement learning is the reward hypothesis[1] – that the optimal policy for any objective can be found by maximiz-

[1] As formulated by Sutton, the reward hypothesis states that: *"all of what we mean by goals and purposes can be well thought of as maximization of the expected value of the cumulative sum of a received scalar signal (reward)".*

ing a scalar reward. Moreover, it has been proposed that a scalar reward is enough to induce behaviors needed to model natural and artificial intelligence (Silver et al., 2021). However, condensing inherently multi-objective problems into a single scalar reward, even when theoretically possible, may be inefficient and limiting (Vamplew et al., 2022). Even if an optimal scalar reward may theoretically exist, designing such a scalar can still be a highly non-trivial task for a practitioner. As demonstrated by various works in multi-objective reinforcement learning (MORL), it is generally useful to decompose complex reward signals into multiple rewards and train separate simpler policies (Hayes et al., 2021; Singh, 1992; Sprague and Ballard, 2003; van Seijen et al., 2017). In robotics, it is often convenient to break down a complex robot into multiple sub-agents, each with its own local policy reacting to changes in its local environment. (Brooks, 1986; Tham and Prager, 1994; Cheng et al., 2021; Li et al., 2021; Van Wyk et al., 2022). Even when tackling a single objective, it can be beneficial to train an ensemble of policies trained using different algorithms (Wiering and Van Hasselt, 2008; Laroche and Feraud, 2017; Song et al., 2023). In general, it is useful to develop modular policies that represent a 'separation of concerns' along different aspects of the overall composite objective (van Seijen et al., 2016).

While it is often easy to break down a problem into modules, building a cohesive solution from multiple alternatives can be much trickier. The problem of modular policy composition has been tackled in various contexts, with numerous proposed solutions including additive and multiplicative mixtures of experts (Hinton, 2002; Urain et al., 2021), voting based ranking (Wiering and Van Hasselt, 2008), winner-take-all approaches (Humphrys, 1995), compositions of utilities or potential fields (Karlsson, 1997; Koren et al., 2000; Urain et al., 2021), and so on. The central idea behind these compositions is to form an average policy from a set of recommendations. However, existing approaches tend to use fairly limited notions of averages, generally restricted to *power means* such as arithmetic means, geometric means, maxima and minima. While these simple averages are easy to implement and understand, it may be desirable to customize the averaging protocol based on the properties of the problem, and desired behavior of the composite policy

In this chapter, we consider the task of averaging policies in its full generality as computing centroids over arbitrary distance spaces. Specifically, we propose embedding individual policies onto task-specific distance spaces where we compute *policy centroids* that minimize distances to individual policies. Naturally, this perspective encapsulates many existing policy compositions. But more importantly, it can serve as a principled framework to design novel compo-

sitions by picking appropriate embedding spaces and corresponding distances. As opposed to averaging policies in their original representations, we can choose to embed them onto spaces that capture important properties of the policies or objectives. Moreover, we also have the flexibility to design distances that utilize the inherent structure of this space, or promote desirable behavior in the composite policy. By choosing an appropriate distance space, or different distance spaces for each objective, policy centroids can extend policy composition beyond simple averages and offer greater control over the process.

10.2 Multi Objective Reinforcement Learning

We tackle policy composition as finding a single-policy solution[2] to multi-objective Markov decision processes (MOMDPs), which are multi-objective generalizations of Markov Decision Processes (MDPs). MDPs provide a general framework for modeling sequential decision making, and form the foundation of many approaches to policy learning, including reinforcement learning (RL). While we mainly focus on multi-objective reinforcement learning (MORL) approaches, much of our analysis should carry over to non-RL approaches as well. Ultimately, we are interested in composing individual policies corresponding to separate objectives into a single composite policy; for the most part, this is agnostic to the algorithms used to learn the individual policies.

[2] There are also various approaches that learn multi-policy solutions, e.g., finding policies that cover the pareto-front with respect to a set of policies. We refer the reader to Hayes et al. (2021) and Roijers et al. (2013) for further details on such methods.

We begin with an overview of MOMDPs, the problem setting for MORL, and two classes of policy composition strategies in MORL. Eventually, we will show that our proposed approach of policy centroids subsume these classes, and allow developing new alternatives.

Multi Objective Markov Decision Processes A MOMDP is a tuple $(\mathcal{X}, \mathcal{A}, T, \gamma, R)$ of state space $\mathcal{X}$, action space $\mathcal{A}$, transition function $T : \mathcal{X} \times \mathcal{A} \times \mathcal{A} \to [0, 1]$, discount factor $\gamma \in [0, 1]$, and a *vector-valued* reward function $R : S \times \mathcal{A} \times \mathcal{X} \to \mathbb{R}^N$. The components r_i of the rewards vector $\mathbf{r}$ are the individual rewards for the N separate objectives. A policy π determines the action executed at a given state; a deterministic policy $\pi(\mathbf{x}) : \mathcal{X} \to \mathcal{A}$ returns a single action, and a stochastic policy $\pi(a|\mathbf{x}) : \mathcal{A} \to [0, 1]$ is a distribution over $\mathcal{A}$. For a state-action pair $(\mathbf{x}, a)$, $Q_i(a|\mathbf{x}) = \mathbb{E}_{\pi_i}(\sum_{t=0}^{\infty} \gamma^t \left([r_i]_t\right) | \mathbf{x}_0 = \mathbf{x}, a_0 = a)$ denotes the state-action value function under the policy π_i for the i-th objective, where $[r_i]_t$ is the i-th reward obtained at the t-th timestep.

Single Policy MORL Single-policy approaches to the MORL problem involve finding a single policy $\pi(\mathbf{x})$ that incorporates all objec-

tives defined by the N reward functions $\{r_i\}$. Note that this is not a well-defined problem with additional context. It may not be possible to find a single policy that simultaneously maximizes the long-term discounted rewards for all objectives. Indeed, for most interesting real-world problems, these objectives will likely be in opposition to one another where maximizing one reward decreases another. In light of potentially conflicting objectives, one strategy to obtain single-policy solutions is to prioritize the objectives by the relative importance, i.e., weights $\{w_i(\mathbf{x}) \geq 0\}$ such that $\sum_i w_i(\mathbf{x}) = 1$. Here, $w(\mathbf{x})$ can be state-dependent, and can either be provided a priori or learned from data. We now discuss two classes of approaches that use these weights to obtain a single policy that incorporates multiple objectives.

Utility Fusion In value-based reinforcement learning, we generally have access to the set of Q-values $\{Q_i(a|\mathbf{x})\}$ that encode the long-term discounted rewards of executing action a at state s with respect to the reward r_i. Thus, the i-th Q-value Q_i defines a utility function that should be maximized by a policy aiming to fulfill the i-th objective. Given relative importance weights $\{w_i\}$, a reasonable approach to MORL is to maximize the weighted average of these Q-values. Unsurprisingly, a popular composition strategy in MORL, known as *utility fusion* (Russell and Zimdars, 2003), treats the weighted arithmetic average of Q-values as the composite Q-value. Formally, the optimal composite policy π^* is defined as follows

$$\pi^*(\mathbf{x}) = \underset{\pi(\mathbf{x})}{\operatorname{argmax}} \sum_i w_i(\mathbf{x}) Q_i^s\left(\pi(\mathbf{x})\right) \tag{10.1}$$

Multiple variants of utility fusion have been proposed in the literature over the years. The GRACIAS algorithm trains the composite-agent as an RL agent maximizing a global reward function (Gupta et al., 2021). This composite-agent learns the weights $w_i(\mathbf{x})$ as its actions. The Distributed Architecture for Mobile Navigation (DAMN) architecture (Rosenblatt, 1997) also uses utility fusion. The hybrid-reward architecture tackles large complex RL problems by (additively) decomposing the overall reward into a sum of rewards (Van Seijen et al., 2017). The authors train separate Q-networks for the constituent rewards, and treat the sum of these constituent Q-functions as the overall Q-function. The constituent Q-functions also share some representation as they are defined as the multi-heads of a neural net, i.e., they share the same lower layers.

While utility fusion is quite intuitive, it is prone to some well-known problems. Firstly, since the arithmetic mean of a set of numbers is naturally sensitive to large outliers, utility fusion can be biased towards objectives with larger Q-values. Secondly, it may also

be problematic in the opposite situation where the disadvantages incurred by one agent are outweighed by the advantages of the majority. Such a *tyranny of the majority* may be undesirable if the disadvantage suffered by the objective in the minority is much larger than advantage gained by individual objectives in the majority. In both issues raised thus far, we see that utility fusion can result in composite policies that may be unfair to certain objectives (Siddique et al., 2020). Finally, when the different Q-functions are trained in isolation without access to the composite Q-function, the composed policy can cause the agent to be stuck in attractors. Under such an *egocentric* training paradigm (Laroche et al., 2017), the composite Q-functions from utility fusion overestimate the value of states where the policies disagree on the optimal action. For large discount factors, such states can become attractors that prevent the agent from making progress towards any objective (Laroche et al., 2017).

Policy Fusion While utility fusion assumes access to utility functions for each objective, often we may instead have access to individual policies instead. In a sense, this represents a more restrictive access model – the utility function holds information about all possible policies as opposed to just the optimal policy. However, this restriction makes the access model more general since the individual policies could be obtained from RL methods that are not value-based, or even non-RL methods from other fields of control theory. We refer to the general class of MORL algorithms that compose multiple policies into a single one as weighted averages as *policy fusion*.

In the setting of deterministic policies, the individual policies simply return actions $\pi(\mathbf{x}) : \mathcal{X} \to \mathcal{A}$. Thus, policy fusion corresponds to defining a composite action as a weighted average of the actions proposed by individual policies. However, such an approach suffers from two critical drawbacks. Firstly, it assumes that $\mathcal{A}$ is a vector space that is closed under weighted addition, or whatever aggregation is used to define weighted averages. If this assumption is not satisfied, the fused-policy may not be feasible[3]. Secondly, even if the action space is closed with respect to addition, it is generally not the case that the weighted averages of actions with high utility also has high utility. Thus, while the individual policies themselves may result in high long-term discounted rewards, the average action may not. In light of these drawbacks of deterministic policy fusion, MORL algorithms generally focus on fusion of stochastic policies where π_i are probability distributions over $\mathcal{A}$.

Given a set of stochastic policies $\{\pi_i\}$ for individual objectives, the composite policy obtained from stochastic policy fusion generally corresponds to a sum-of-experts or a product-of-experts model

[3] For example, if the action space is $\mathcal{A} = \{\uparrow, \downarrow, \leftarrow, \rightarrow\}$, weighted averages of actions might result in diagonal directions, which cannot be executed by the agent.

defined as follows

$$\text{Sum of Experts (SoE):} \qquad \bar{\pi} \propto \sum_i w_i(\mathbf{x})\pi_i(a|s) \tag{10.2}$$

$$\text{Product of Experts (PoE):} \qquad \bar{\pi} \propto \prod_i w_i(\mathbf{x})\pi_i(a|s). \tag{10.3}$$

While not always described in such terms, these compositions are equivalent to element-wise arithmetic and geometric means of the individual policies, which are special cases generalized f-means, also called Kolmogorov or quasi-arithmetic means (Bullen, 2013).

A common strategy in prior work has been to train individual policies $\{\pi_i\}$ in isolation and train weights $\{w_i\}$ as a meta-controller or a composite agent. Prior works have treated this composite agent as an RL agent learning to pick between the sub-policies (Lin, 1993; Simpkins and Isbell, 2019). Simpkins and Isbell (2019) use this approach to develop the Arbi-Q policy-composition algorithm that is robust against mismatched rewards scales between the various sub-agents. These approaches can be interpreted as minimizing the policy-fusion loss function where the vector of weights (w_i) only has one non-zero entry. Recently, Qureshi et al. (2019) presented an algorithm where stochastic sub-policies are represented as Gaussian distributions $\mathcal{N}(\mu_i, \sigma_i)$, and the overall policy is their weighted combination $\sum_i w_i \mathcal{N}(\mu_i, \sigma_i)$.

The various instantiations of utility fusion and policy fusion we have described can be understood as performing some form of arithmetic or geometric mean of utilities or policies. We now define *policy centroids* as our proposed framework that includes both utility fusion and policy fusion as sub-classes. Additionally, we demonstrate how policy centroids allow us to expand the scope of these composition classes, and also introduce additional policy composition strategies.

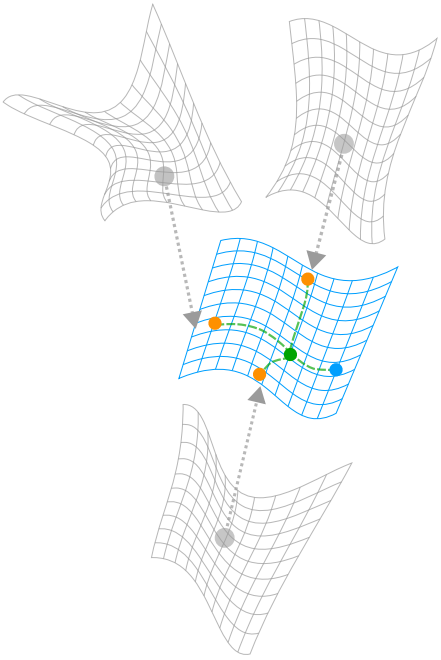


Figure 10.1: **Visualizing Policy Centroids** Policy centroids correspond to the weighted centroid of multiple policies, where the distance with respect to a policy (blue) is computed in its own distance space. Other policies (orange) are mapped onto this distance space via policy-specific embedding functions.

10.3 Policy Centroids

Consider a set $\{\pi_i\}_{i=1}^N$ of N policies, such that each policy π_i has been formulated separately based only on the i-th component of the reward function. These policies must now be combined into a single composite policy $\bar{\pi}$, without assuming access to a composite reward function or a known decomposition of the composite objective into individual objectives. Let $(\mathcal{M}_i, d_i)$ be distance spaces[4] where π_i can be compared with alternatives via distances d_i. Policies are mapped onto these spaces using policy embeddings $\mu_i(\pi)$. Finally, given a set of optional non-negative weights $\{w_i\}_{i=1}^T$ with $\sum_i w_i = 1$, the composite policy $\bar{\pi}$ or the *policy centroid* is defined as follows

$$\bar{\pi} = \underset{\pi}{\operatorname{argmin}} \sum_i w_i d_i(\mu_i(\pi_i),\ \mu_i(\pi)). \tag{10.4}$$

[4] A distance space $(\mathcal{M}, d)$ is a set $\mathcal{M}$ equipped with a non-negative, symmetric, and reflexive distance function $d(\mathbf{x}, \mathbf{x}')$ for $\mathbf{x}, \mathbf{x}' \in \mathcal{M}$.

To design a policy centroid, we pick (1) embedding functions μ_i that capture important features of policies, and (2) distances d_i that account for meaningful differences between them. The weights w_i are linear relative importances of the T objectives. composite objective or learned with respect to a composite reward; otherwise, they are set to be uniformly distributed.

Policy centroids provide additional means of affecting properties of the composite policy through the choice of the distance spaces $(\mathcal{M}_i, d_i)$. In the following sections, we show how various existing compositions can be interpreted as policy centroids. Moreover, we provide examples where different choices of embeddings μ_i and distances d_i can lead to composite policies with useful properties, and extend existing compositions to new problem settings.

10.4 Policy Centroids for Utility Fusion

Utility Centroids Given a set of Q-functions from different policies, the objective function for utility fusion in Equation 10.1 maximizes the total (equivalently average) utility over all objectives. We can recast this optimization as a policy centroid in Equation 10.4 as follows

$$\text{Utility Fusion:} \quad \bar{a} = \operatorname*{argmin}_a \sum_i w_i \left(Q_i(a_i^*|\mathbf{x}) - Q_i(a|\mathbf{x})\right) \tag{10.5}$$

where $a_i^* = \text{argmax}_{a'} Q_i(\mathbf{x}, a')$ is the recommended action for the i-th objective. Here, task-specific embeddings are simply the Q-values $\mu_i(a|\mathbf{x}) = Q_i(a|\mathbf{x})$, and the distance function is just the difference in Q-values[5].

We can generalize Equation 10.6 to some extent by introducing a new parameter $\beta > 0$ that provides addition control[6] over the composition, resulting in the following *utility centroids*

$$\text{Utility Centroids:} \quad \bar{a} = \operatorname*{argmin}_a \sum_i w_i (Q_i(a_i|s) - Q_i(a|\mathbf{x}) + 1)^\beta \tag{10.6}$$

where, the distances are defined as $d_i(Q_i(a_1|\mathbf{x}), Q_i(a_2|\mathbf{x})) = (|Q_i(a_1|\mathbf{x}) - Q_i(a_2|\mathbf{x})| + 1)^\beta - 1$. Equation 10.6 minimizes the average disadvantage or opportunity cost $(Q_i(a_i|\mathbf{x}) - Q_i(a|\mathbf{x}) + 1)^\beta$ of picking a over the preferred $a_i = \text{argmax}_a Q_i(a|\mathbf{x})$. Note that with $\beta = 1$, the resulting policy is the same as that for utility fusion in Equation 10.1 and Equation 10.5.

As illustrated in Figure 10.2,the exponent β plays a similar role to weights w_i ; while w_i stretch or squeeze distances between policies linearly, β magnifies or minimizes distances between policies exponentially. Setting $\beta > 1$ (resp. $\beta < 1$) magnifies (resp. minimizes) differences in disadvantages leading to sharper (resp. flatter) dis-

[5] This distance is non-negative since $Q_i(a^*|\mathbf{x}) > Q_i(a|\mathbf{x})$ by the definition of Q values.

[6] The optional offset +1 simply sets the domain of the exponential to $\mathbb{R}_{\geq 1}$, avoiding the inverse behavior of the exponent for $(0, 1)$.

advantage curves. However, unlike weights w_i, we can set the same exponent β across all policies and still affect relative distances.

In Figure 10.3, filled-circles of the same color correspond to the recommended actions from four policies for a given choice of β. When $\beta = 1$ (green), the policy recommending action a_4 has a slightly lower disadvantage, pushing the centroid closer to a_4. With $\beta > 1$ (orange) differences in disadvantages are magnified, pushing the centroid further towards a_4. With $\beta < 1$ (purple), the small difference in disadvantages is minimized, and the centroid is not biased towards any one policies. With $\beta = 1$, Equation 10.6 reduces to standard utility fusion. Thus, β can serve as a useful hyperparameter, especially in the absence of non-uniform weights w_i. As we now discuss, adjusting β can alleviate some known problems with utility fusion in such settings.

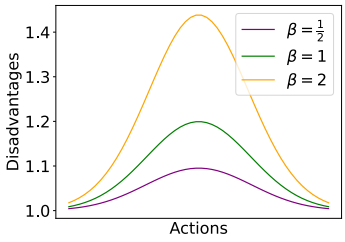


Figure 10.2: **Varying β in Utility Centroids** Visualizing the effect of varying β in Equation 10.6.

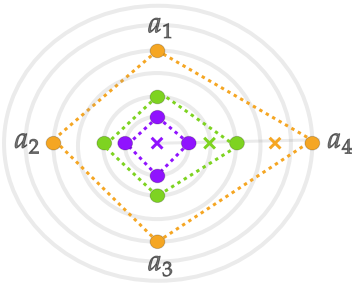


Figure 10.3: **Illustration of Utility Centroids with Varying β** Visualizing the effect of varying β in Equation 10.6.

Preventing Attractors When the Q-values of multiple policies are trained *egocentrically* (without taking into account the composite Q-value), utility fusion with uniform weights w_i overestimates the value of states where policies disagree on the optimal action (Laroche et al., 2017). For large discount factors γ, these states can become attractors where the agent gets stuck; restricting us to myopic sub-optimal policies with low discount factors.

In Equation 10.6, reducing β essentially minimizes disagreement between policies and can help us escape attractors. To demonstrate, we consider the two examples from Laroche et al. (2017) illustrating attractors in utility fusion. In the first example (Figure 10.4), we consider a 3 state deterministic MOMDP where the agent in state $\mathbf{x}_0$ can stay in place with action a_0, or move to one of two terminating goal states $\mathbf{x}_1$ and $\mathbf{x}_2$ with actions a_1 and a_2 respectively. We define separate objectives with reward $r > 0$ for reaching $\mathbf{x}_1$ and $\mathbf{x}_2$ respectively, and no reward for staying in place. In the second example in Figure 10.5, the PacBoy agent is equidistant from a reward in either of the three directions $\{\uparrow, \leftarrow, \rightarrow\}$ in a simple deterministic MOMDP; choosing $\downarrow$ keeps it in place. In both examples, egocentric utility fusion with uniform weights w_i gets stuck when $\gamma > 0.5$ (Laroche et al., 2017).

In Figures 10.6(a) and 10.6(b), we plot the combinations of γ and β for which the policy *centroid* in Equation 10.6 is stuck (shaded black) and for which it reaches one of the two goal states (shaded white). As we decrease β, we can allow larger discount factors while still avoiding getting stuck. However, using too small a β may also lead to undesirable behavior where the relative preferences of policies are ignored.

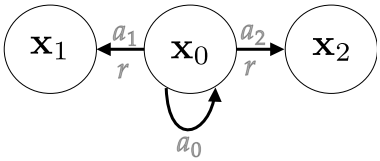


Figure 10.4: **Illustrations of MDPs with attractor states.** The MDP defined in Laroche et al. (2017) exhibiting attractors.

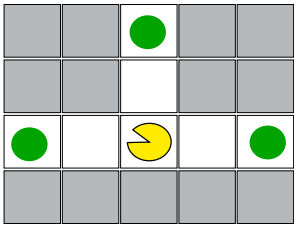


Figure 10.5: **The PacBoy Example** The PacBoy agent (yellow) surrounded by equidistant goals (green).

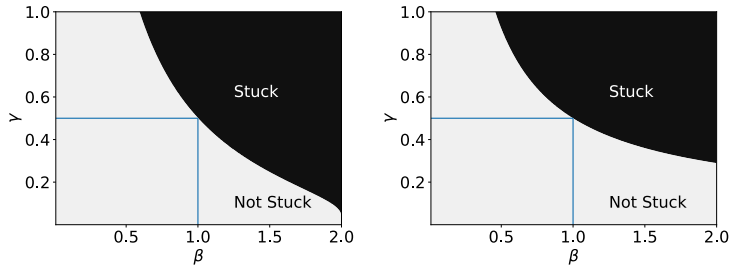


Figure 10.6: **Utility centroids to escape attractors**. The blue lines are reference markers for utility fusion, or equivalently utility centroids ($\beta = 1$).

Preventing the tyranny of the majority Utility fusion can also suffer from a tyranny of the majority, where an objective dominated by a large number of competing objectives may never be fulfilled. If the minority objective is safety critical, it may be desirable to forego the other goals to prevent an accident. If the objectives involve human stakeholders, this majority bias can be unfair – the minority stakeholder may lose confidence in the system if their objectives are consistently ignored in service of the majority.

As an illustration, consider a set of N objectives $\{O_N\}_{i=1}^N$ defined over a MOMDP with two actions $\{a_1, a_2\}$; O_N corresponds to avoiding accidents, while the others correspond to reaching various goals. At some state, O_1 identifies action a_2 as unsafe with a disadvantage of d. In contrast, a_2 is slightly favorable over a_1 for the goal-seeking objectives, assigning disadvantage $\epsilon < d$ for a_1. With only two tasks $\{O_1, O_2\}$, utility fusion opts for the safe action a_1 as it has lower disadvantage. However, as we add other objectives to the mix, the unsafe action a_2 is picked when $n > d/\epsilon$. With the utility centroid in Equation 10.6, we can increase β to intensify individual preferences, leading to a larger threshold. In Figure 10.4, we plot various combinations of β and n for which the agent picks the unsafe action a_2 (shaded black) and the safe action a_1 (shaded white) when $\epsilon = 0.1d$, i.e., each goal-seeking agent only favors the unsafe action by a small margin of 10%. However, as β becomes very large, the minority may exert too much influence, preventing the agent from ever achieving its other goals. Thus β will likely have to be tuned to balance these effects.

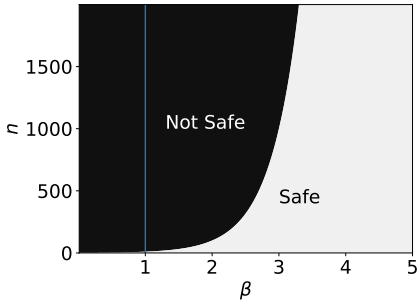


Figure 10.7: **Utility centroids to prevent the tyranny of the majority**. The blue lines are reference markers for utility fusion, or equivalently utility centroids ($\beta = 1$).

10.5 Policy Centroids for Stochastic Policy Fusion

We now turn to the problem of composing stochastic policies represented as probability distributions $\pi(a|\mathbf{x})$ over actions. Such policies can arise, for instance, due to exploration or as solutions to partially observable MDPs or adversarial games.

Generalized Means of Experts As discussed in Section 10.2, two common approaches of composing stochastic policies is via sums (Equation 10.2) and products (Equation 10.3) of experts. While not always described as such, these compositions are equivalent to element-wise arithmetic and geometric means, which are special cases of generalized f-means, which, in turn, are instances of Fréchet centroids (Bullen, 2013). We can thus formulate policy centroids corresponding to generalized-means-of-experts (GMoE) through a monotonic, continuous and invertible function f as follows:

$$\begin{aligned} \bar{\pi} &\propto \operatorname*{argmin}_{\pi} \sum_i d^2(\pi_i, \pi) \\ \text{where} \quad d^2(\pi, \pi') &= \sum_a |f(\pi(a|\mathbf{x})) - f(\pi'(a|\mathbf{x}))|^2. \end{aligned} \tag{10.7}$$

This centroid defines the well-known class of generalized f-means. With $f(x) = x^\beta$ (for $\beta \neq 0$), we get the family of *power means*, which includes sum-of-experts (arithmetic mean for $\beta = 1$) and product-of-experts (geometric mean for $\beta \to 0$), as well as a spectrum of other means.

10.5.1 Implicit Stochastic Policy Fusion

While GMoEs are intuitive and ubiquitous, they may not always be ideal, for instance, when composing *implicit* stochastic policies. Implicit policies $\hat{\pi}$ are distributions without a known density function, but allow sampling. Such policies can be useful, for instance, to incorporate unknown noise distributions or as a flexible policy class for complex action distributions (Tang and Agrawal, 2018). Given sets of actions $\{a_j^{(i)} \sim \hat{\pi}_i\}$ drawn from a set of implicit policies $\{\hat{\pi}_i\}$, we now wish to generate new actions $\{\bar{a}_m\}$ from the composite policy $\bar{\pi}$.

As a concrete example, consider the following sum-of-experts composition problem where we wish to evaluate the expected value of some function $g(a)$ under the distribution of the composite policy. Assuming that both the function g and individual policies are computationally expensive to query, we seek a good estimate of $\mathbb{E}_{\bar{\pi}}\, g(a)$ with minimal samples from individual policies as well as the composite policy. SoEs are not directly applicable since they require the policies' probability densities. We would thus need some form of density estimation, which can be sample intensive for high-dimensional actions. With policy centroids, we can extend sum-of-experts to this new problem setting. Specifically, we can embed implicit policies onto reproducing kernel Hilbert spaces, and use the maximum mean discrepency between distributions as our distance function.

MMD Policy Centroids The maximum mean discrepancy (MMD) is an integral probability (pseudo) metric (IPM) between probability measures that captures the maximal difference between expectations of functions (Gretton et al., 2012). For a set of functions $\mathcal{G}$, the IPM between distributions π and π' is the supremum of the difference in expectations $\mathbb{E}_\pi(g) - \mathbb{E}_{\pi'}(g)$ over all $g \in \mathcal{G}$. We obtain the MMD when $\mathcal{G}$ is the unit-ball in a reproducing kernel Hilbert space (RKHS) (Gretton et al., 2012).

For any symmetric positive-definite kernel $k(\cdot, \cdot)$ on a domain $\mathcal{A} \times \mathcal{A}$, there exists a corresponding RKHS $\mathcal{H}_k$ and k is the canonical feature mapping to $\mathcal{H}_k$. Additionally, we can use the kernel to also map probability distributions π defined over $\mathcal{A}$. These embedded distributions $\boldsymbol{\mu}_\pi \in \mathcal{H}_k$, known as *kernel mean embeddings*, are defined as $\boldsymbol{\mu}_\pi = \int k(a, \cdot) d\pi(a)$. The empirical estimates of these embeddings $\hat{\boldsymbol{\mu}}_\pi = \frac{1}{n}\sum_{j=1}^{n} k(a_j \sim \pi, \cdot)$ converge in MMD to $\boldsymbol{\mu}_\pi$ at a rate of $O(n^{-1/2})$ (Gretton et al., 2012), which is independent of the dimensionality of $\mathcal{A}$ (in contrast to the curse of dimensionality for density estimation) (Gretton et al., 2012). The MMD[7] between two kernel mean embeddings is the $\mathcal{H}_k$-norm of their difference:

$$\text{MMD}(\hat{\pi}, \hat{\pi}') = || \hat{\boldsymbol{\mu}}_\pi - \hat{\boldsymbol{\mu}}_{\pi'} ||_{\mathcal{H}_k}$$

[7] Additionally, if the kernel is *characteristic* (e.g. Gaussian, Laplace), the MMD is a metric such that $\text{MMD}(\pi, \pi') = 0 \Leftrightarrow \pi = \pi'$ (Gretton et al., 2012).

Given a kernel k, we define *MMD policy centroids* that compare empirical policies with respect to the MMD as follows:

$$\text{MMD Policy Centroid} \qquad \bar{a} \sim \bar{\pi} = \operatorname*{argmin}_{\hat{\pi}} \sum_i w_i \text{MMD}(\hat{\pi}_i, \hat{\pi}) = \operatorname*{argmin}_{\hat{\boldsymbol{\mu}}} \sum_i w_i ||\hat{\boldsymbol{\mu}}_i - \hat{\boldsymbol{\mu}}||^2_{\mathcal{H}_k}.$$

Note that the above objective is essentially what is optimized by the kernel herding algorithm (Chen et al., 2010) that draws samples from a kernel mean embedding; here, the target is instead the weighted sum $\sum_i w_i \hat{\boldsymbol{\mu}}_i$. Using kernel herding, we can draw the t-th sample $\bar{a}_t$ from $\bar{\pi}$ as follows

$$\bar{a}_t = \operatorname*{argmin}_a \sum_{i,j} -2 w_i k(a_j^{(i)}, a) + \mathcal{I}(t > 1) \sum_{m=1}^{t-1} \frac{1}{t} k(\bar{a}_m, a) \tag{10.8}$$

where $\mathcal{I}(t > 1)$ is the indicator function. While the first term in the objective encourages $\bar{a}_t$ to be similar to the set of samples $\{a_j^{(i)} \sim \hat{\pi}_i\}$, the second term encourages it to be *dissimilar* to samples $\{\bar{a}_m\}_{m=1}^{t-1}$ generated in prior iterations. This latter repulsive force improves sample diversity by inducing negative autocorrelations. For bounded kernels corresponding to finite dimensional $\mathcal{H}_k$, the herded *super* samples converge in MMD at a rate of $O(n^{-1})$ to the target distribution – faster than $O(n^{-1/2})$ for iid sampling from the true distribution (Chen et al., 2010). Even though $\mathcal{H}_k$ is generally not finite, we

still tend to get fast convergence in practice – even if Equation 10.8 is only solved approximately (Chen et al., 2010).

Since the fast convergence of also carries over to expectations of functions $\mathbb{E}_{\pi} g(a) = \frac{1}{M} \sum_{m=1}^{M} g(\bar{a}_m)$, MMD centroids are well-suited for our purposes – to compute sample based expectations over policies using minimal samples. Firstly, using the herded super samples, we should require fewer evaluations of g before $\mathbb{E}_{\bar{\pi}}(g)$ converges. Secondly, since we bypass an explicit density estimation, we should need fewer samples from individual policies as well. Moreover, since the convergence rate of the *implicit* density estimation $\hat{\boldsymbol{\mu}} \approx \boldsymbol{\mu}$ is independent of the dimensions of $\mathcal{A}$, MMD centroids should scale better to high-dimensional action spaces.

To demonstrate the advantages of MMD centroids, we consider a set of 3 Gaussian policies $\{\pi_i = \mathcal{N}(a_i, 0.1\mathbb{I})\}$ with mean actions $a_i \in \mathbb{R}^d$. Assuming uniform policy weights w_i, the composite distribution for the true weighted sum-of-expert policy is simply $\sum_i \frac{1}{3}\mathcal{N}(a_i, 0.1\mathbb{I})$. Our goal is to generate M samples $\{\bar{a}_m\}_{m=1}^{M}$ from the composite policy (only using samples from π_i) and use them to compute expected values of functions. Here, we fix the target function to be the first moment (mean action) of the composite policy, which is simply $\frac{1}{3}a_i$.

We compare estimation errors for traditional sum-of-expert, MMD centroids, and iid samples from the true composite policy. For MMD centroids, we use the Gaussian kernel with its bandwidth set using the median-trick heuristic, and generate samples via kernel herding. Since the true composite policy is generally unknown, we avoid tuning hyperparameters for the optimization in Equation 10.8 by opting for a brute force search within a random subset of M action samples drawn from individual policies. For the sum-of-expert composition (Equation 10.2), we draw samples from the kernel density estimate of the composite policy using a Gaussian kernel (with the median-trick bandwidth).

We present the results of our experiments in Figure 10.8, Figure 10.9 and Figure 10.10, where we fix $(d = 32, M = 250)$, $(M = 250, N = 32)$ and $(d = 32, N = 32)$ respectively and plot errors with respect to varying N (samples from individual policies), d (dimensionality of actions), and M (samples from the composite policy). As the results demonstrate, MMD centroids offered better scaling with respect to all three parameters tested. Thus, using the policy centroid framework, we are able to adapt the traditional sum-of-expert composition strategy to implicit stochastic policies.

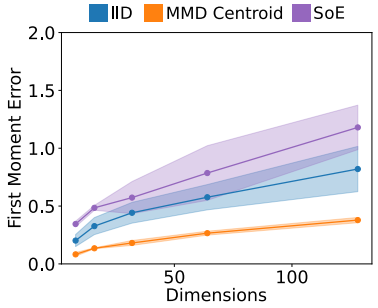


Figure 10.8: **Errors for MMD centroids and sum-of-expert composition as a function of dimensionality of $\mathcal{A}$.** Error bars are standard deviations across 3 random seeds

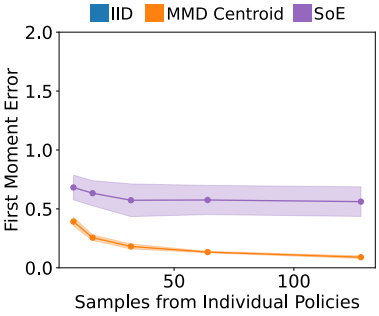


Figure 10.9: **Errors for MMD centroids and and sum-of-expert composition as a function of samples from the composite policy.** Error bars are standard deviations across 3 random seeds

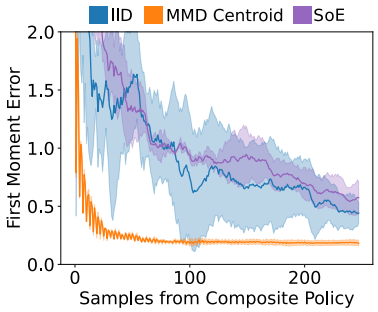


Figure 10.10: **Errors for MMD centroids and sum-of-expert composition as a function of samples from the composite policy**. Error bars are standard deviations across 3 random seeds

10.5.2 Riemannian Motion Policies

We have thus far focused on two policy compositions often encountered in multi-objective RL. We now turn our attention to a different setting within the framework of Riemannian motion policies (RMPs) from robotics, designed to combine multiple acceleration based policies for cohesive motion generation (Cheng et al., 2021). As described below, policy centroids share the intuition behind RMP-composition in using task-specific distance spaces, and can also extend RMP-composition to the setting of stochastic controllers.

RMP Composition While a robot's complete state is defined on a configuration space $\mathcal{Q}$, its various objectives can be defined on separate task spaces $\mathcal{X}_i$. Configuration space states $\mathbf{q}$ are mapped onto task-space states $\mathbf{x}_i$ via task-maps $\phi_i : \mathcal{Q} \to \mathcal{X}_i$. Using $\mathbf{J}_i$, the Jacobians of the task-maps ϕ_i, we can also map configuration space velocities and accelerations into their task-space counterparts as

$$\dot{\mathbf{x}}_i = \mathbf{J}_i \dot{\mathbf{q}} \;\text{ and }\; \ddot{\mathbf{x}}_i = \mathbf{J}_i \ddot{\mathbf{q}} + \dot{\mathbf{J}}_i \dot{\mathbf{q}}$$

A task-space RMP over $\mathcal{X}_i$ is a tuple $(\mathbf{f}_i, \mathbf{M}_i)$ indicating a recommended force $\mathbf{f}_i = \mathbf{M}_i \ddot{\mathbf{x}}_i$ given a positive semi-definite inertia matrix $\mathbf{M}_i$.

The general idea behind the RMP framework is to combine task-space RMPs defining desired accelerations $\ddot{\mathbf{x}}_i$ into a composite RMP in the configuration space. Given a set of task-space RMPs $\{(\mathbf{f}_i, \mathbf{M}_i)\}$, the composite configuration-space RMP $(\mathbf{f}, \mathbf{M})$ is computed by *pulling-back* the task-space forces into the configuration space as

$$\texttt{pullback}(\mathbf{f}_i) = \mathbf{J}_i^T(\mathbf{f}_i - \mathbf{M}_i \dot{\mathbf{J}}_i \dot{\mathbf{q}}_i) \;\text{ and }\; \texttt{pullback}(\mathbf{M}_i) = \mathbf{J}_i^T \mathbf{M}_i \mathbf{J}_i$$

The composite force and inertia matrices are then computed by adding up the pulled-back forces $\mathbf{f} = \sum_i^T \texttt{pullback}(\mathbf{f}_i)$ and inertia matrices $\mathbf{M} = \sum_i \texttt{pullback}(\mathbf{M}_i)$. The composite control or acceleration $\ddot{\mathbf{q}} = \mathbf{M}^+ \mathbf{f}$ is the solution to the following objective function

$$\underset{\ddot{\mathbf{q}}}{\text{argmin}} \; \frac{1}{2} \sum_{i=1}^{T} \|\mathbf{J}_i \ddot{\mathbf{q}} + \dot{\mathbf{J}}_i \dot{\mathbf{q}} - \ddot{\mathbf{x}}_i\|^2_{\mathbf{M}_i} = \underset{\ddot{\mathbf{q}}}{\text{argmin}} \; \frac{1}{2} \sum_{i=1}^{T} \|\mu_i(\ddot{\mathbf{q}}) - \mu_i(\ddot{\mathbf{x}}_i)\|^2_{\mathbf{M}_i}. \tag{10.9}$$

In the second equality above, we have rewritten the RMP composition objective similar to a policy centroid with $\mu_i(\ddot{\mathbf{q}}) = \mathbf{J}_i \ddot{\mathbf{q}} + \dot{\mathbf{J}}_i \dot{\mathbf{q}}$, where $\mu_i(\ddot{\mathbf{x}}_i) = \ddot{\mathbf{x}}_i$. Each term in this objective can be interpreted as a task-specific distance (defined via $\mathbf{M}_i$) between $\mu_i(\ddot{\mathbf{q}})$ and desired task-controls $\ddot{\mathbf{x}}_i$. Thus, policy centroids share RMP's underlying idea of employing task-specific mappings and computing centroids with

respect to task-specific distances. However, policy centroids extend this idea beyond acceleration-based controls to generic settings including, for instance, discrete action spaces. Moreover, as we now discuss, the policy centroid perspective allows us to extend RMP composition to the case of stochastic controllers.

Wasserstein Policy Centroids The RMP framework assumes deterministic controls $\ddot{\mathbf{x}}_i$ or $\mathbf{f}_i$ and is not directly applicable when these controls are stochastic. Incorporating stochasticity into RMPs can be useful, for instance, when individual controllers are learned via RL with exploration, or to escape local stationary points where competing controls cancel each other out. To compose such stochastic RMP controllers, we can collect samples of desired controls from each individual RMP and pull them back onto the configuration space.

As in Equation 10.9, the composite metric $\mathbf{M}$ defines a distance measure for pulled-back acceleration controls. In the deterministic setting with only one sample per RMP, we would essentially obtain the composite control formed via the metric-weighted averaging in Equation 10.9. But in the stochastic setting, we need to perform an average both in the space of controls (as in deterministic RMPs), but also in terms of probability masses spread across controls. Essentially, we require some form of stochastic averaging that can take into account the ground distance between controls in the pulled-back space, while also performing a probabilistic average. The family of Wasserstein distances between distributions is exactly such a statistical distance.

The p-th Wasserstein distance $\mathcal{W}_p(\pi, \pi'; d)$ between two distributions π and π' defined with respect to a ground metric $d(\cdot, \cdot)$ represents the cost of transforming π into π' by moving probability mass in the ground space of samples; the cost of moving probability mass is measured using d. Using a Wasserstein distance as our statistical distance between policies, we can define Wasserstein policy centroids for RMPs with stochastic controllers as the Wasserstein barycenter of the individual pulled-back control distributions. Similar to how RMP composition computes sums or metric-weighted averages of controls, the Wasserstein policy centroids would allow us to average entire distributions over controls.

$$\bar{\pi} \propto \operatorname*{argmin}_{\pi} \sum_i \mathcal{W}_p^2(\pi_i, \pi;\ d_i(\cdot)) \tag{10.10}$$

10.6 Conclusion

We presented a framework for modular policy composition, viewing the problem as computing centroids over distance spaces. Besides

subsuming various commonly used composition strategies, policy centroids also provide a principled way to develop new compositions by identifying appropriate policy embeddings and metrics. As an illustration, we developed utility centroids and MMD centroids, and demonstrated their advantages over existing approaches. Moreover, we can idenfity various avenues of future research to extend the present work. For instance, we have thus far ignored any relationships between state and action spaces of objectices. As with RMPs, we can organize such interrelated objectives via hierarchical network structures. It may be valuable to incorporate this structure into the composition framework as well. Furthermore, instead of hand-crafting metrics, we could potentially learn them directly from data. This would essentially generalize current approaches that learn importance weights for policies. Instead of learning weights that linearly scale distances between policies, we may be able to learn more sophisticated transformations of policy spaces.

Part V

Conclusion

11
Conclusion

In this thesis we have presented multiple methods of that incorporate metric-structure into specific machine learning problems spanning various domains. Specifically, we tackled the problems of learning quantum-inspired probabilistic models, drawing samples over Riemannian manifolds using kernel herding, learning locality-preserving representations in reinforcement learning, and composing policies in multi-objective reinforcement learning. For each of these problems, we identified specific metric structures in data or parameter spaces, and presented our proposal of exploiting this structure to improve existing algorithms. We now return to the original research question we laid out at the beginning of this thesis: *how can we utilize metric structure in machine learning algorithms to improve their performance?* We now evaluate how the contributions provided in this thesis answer this question within specific problem domains.

Metric Informed Quantum Graphical Models

> How can we utilize *the Stiefel manifold* structure of model parameters in *hidden Quantum Markov models* to ensure *feasibility and improve accuracy*.

By leveraging the Stiefel manifold parameterization of hidden quantum Markov models (HQMMs), we proposed an efficient learning method for HQMMs based on Riemannian gradient descent (Adhikary et al., 2020). While prior work had also identified this Riemannian parameterization, it relied on optimization through sequential composition of Givens rotations, utilizing the invariance of orthonormality with respect to rotations (Srinivasan et al., 2018c). Although this approach ensured feasibility of learned parameters, it was computationally expensive and prohibitively slow for large HQMMs. This restricted prior applications of HQMMs to relatively small problems since the the number of parameters in HQMMs grows rapidly

with both the latent state dimension and observation space. We improved upon this approach by employing retractions on the Stiefel manifold, to constrain optimization paths to geodesics on the manifold (Algorithm 1). Through evaluations on both synthetic and real world data, we demonstrated that our approach is not only faster, but also provides greater predictive accuracy.

In addition to providing a learning algorithm for HQMMs, we also extensively analyzed the expressiveness of these models. While prior results had established HQMMs as more expressive than HMMs, we expanded this analysis to include other classical generalizations of HMMs from machine learning including observable operator models (OOMs), predictive state representations (PSRs), norm-observable operator models (NOOMs), stochastic weighted automata (SWA), as well as uniform variants of multiple tensor network models from quantum information including locally purified states (LPS) and Born machines (BMs). Through a series of novel expressiveness relationships, we established a hierarchy of linear sequential systems with respect to HQMMs, and established HQMMs as the most expressive model class within those considered that can relaibly avoid the negative probability problem (Figure 7.6). We also established a similar hierarchy of models in the controlled setting, and demonstrated how HQMMs subsume various quantum generalizations of controlled linear sequential systems (Figure 6.1).

Metric Informed Sampling

How can we utilize *the Riemmanian manifold* structure of data spaces in *kernel herding* to ensure *feasibility of samples and improve convergence to target distributions*.

We introduced a method for extending kernel herding to Riemannian manifolds by leveraging (1) geometry-aware kernels that respect the intrinsic distance of the manifold, and (2) Riemannian optimization to ensure that sampled points remain on the manifold. We evaluated our approach on two tasks involving sampling from distributions supported on Riemannian manifolds. In the first task, we showed improved convergence when resampling from manifold-supported distributions, outperforming both heuristic projection-based herding and importance resampling using optimal transport maps. In the second task, we integrated our method into kernel recursive approximate Bayesian computation to estimate parameters of simulators defined on the symmetric positive-definite manifold, achieving higher accuracy than both heuristic projection methods and the commonly used Cholesky decomposition parameterization. The

results on these experiments support our hypothesis that explicitly incorporating Riemannian geodesic distances into kernel herding results in improved sample convergence to manifold-supported target distributions.

Metric Informed Reinforcement Learning: BeigeMaps

How can we utilize *bisimulation metrics* in *behavioral representation learning* algorithms in RL improve *policy performance*.

We introduced an approach to learning novel representations based on behavioral metrics that allow distortions in long-range distances, while preserving *local* metric structure – inducing representations that highlight natural value-based clusters in the state space. These new representations, called behavioral eigenmaps (BeigeMaps), correspond to learned neural eigenfunctions of Laplacian eigenmaps with respect to behavioral distances. We evaluated the utility of BeigeMaps by jointly learning representations with policies on the Deep Mind control benchmark (Tassa et al., 2018) with high dimensional image observations. Our experiments demonstrated that incorporating BeigeMaps as a drop-in component into existing behavioral distance learning algorithms led to improved policy performance.

Metric Informed Reinforcement Learning: Policy Centroids

How can we utilize *policy-induced distance spaces* for *policy composition* in multi-objective RL to induce *desirable composition properties*.

We presented a framework for modular policy composition, viewing the problem as computing centroids over policy-induced distance spaces. Besides subsuming various commonly used composition strategies, policy centroids also provide a principled way to develop new compositions by identifying appropriate policy embeddings and metrics. With utility centroids, we incorporated existing utility fusion approaches as policy centroids, with additional parameters that can be tuned to mitigate known problems such as the formation of attractors, and undesirable behaviors corresponding to “the tyranny of the majority". For implicit stochastic policies, we introduced MMD-policy-centroids with the maximum mean discrepancy (MMD) as the underlying metric. We demonstrated that MMD centroids can offer faster convergence to the target composite policy than existing sum-of-expert compositions. Finally, we demonstrated how Riemannian motion policies (RMPs) can also be viewed as policy centroids, and

how they can be expanded to the setting of implicit stochastic policies via Wasserstein policy centroids. Overall, we demonstrated how policy centroids provide a flexible framework for geometry-aware policy composition that can not only subsume various existing composition techniques, but also provide a new means of inducing useful properties in composite policies.

Part VI

Appendix

12
Appendix: Metric Informed Quantum Graphical Models

12.1 Random Initialization

The ROSM algorithm begins with an initial guess of the optimal parameters $\boldsymbol{\kappa}$ and a random intial density matrix $\boldsymbol{\rho}$. By 'burning-in' a reasonable number of initial entries in sequences, we minimize the effect of randomly initializing $\boldsymbol{\rho}$. To investigate the sensitivity of ROSM to initializations of $\boldsymbol{\kappa}$, we trained models on the synthetic HQMM and HMM datasets over 3 random seeds. As shown in the results in Figure 12.1, ROSM 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 12.1.

12.2 Hyperparameter Selection

To facilitate a clear comparison with GS, we used the same batch size as in Srinivasan et al. (2018e), 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.

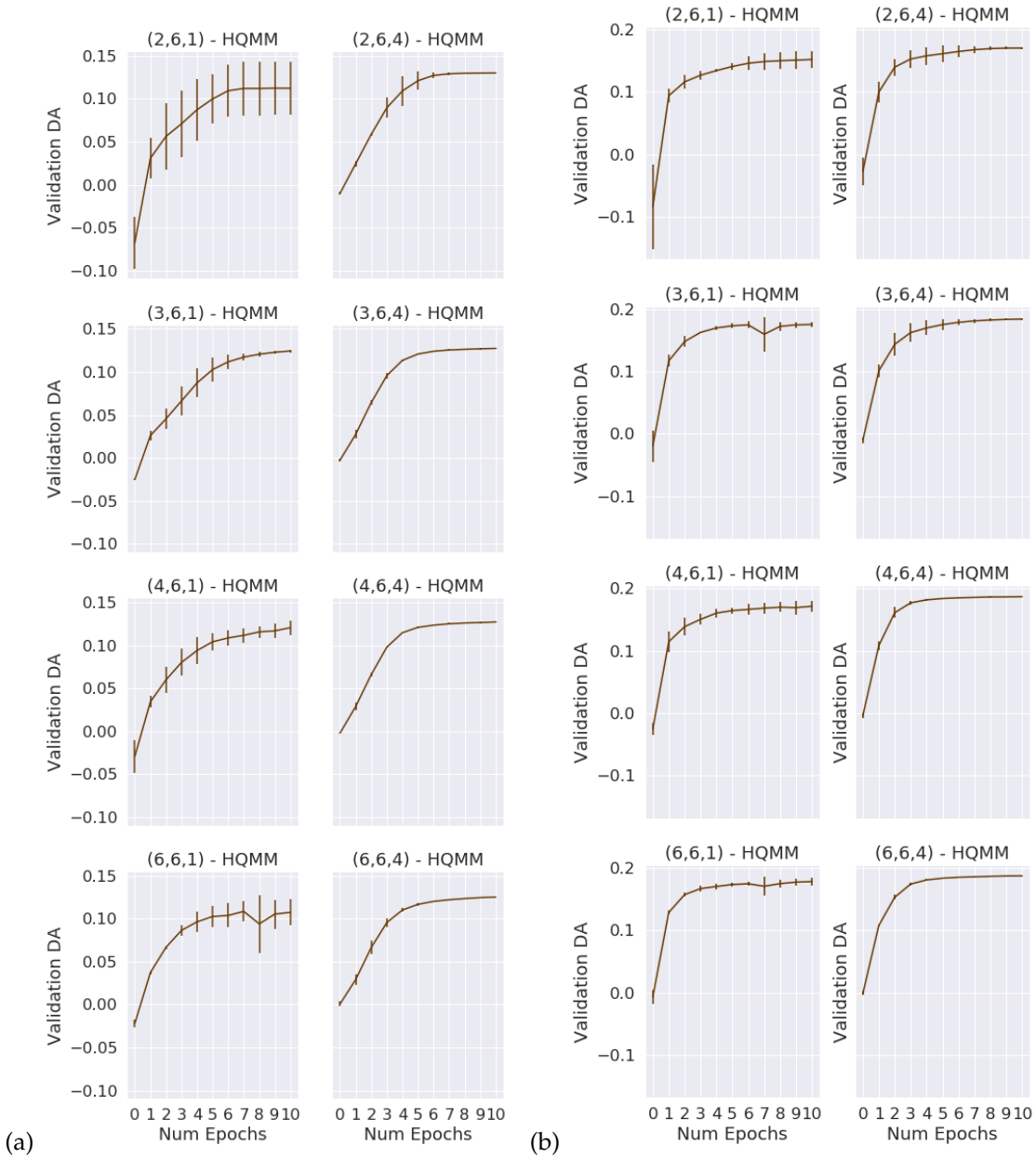


Figure 12.1: **ROSM's Sensitivity to Random Initializations of κ** Validation set accuracies obtained across 10 epochs for HQMMs trained on 3 different random initializations on the synthetic HQMM dataset (a) and synthetic HMM dataset (b). ROSM is sensitive to κ initialization for the smallest models, but is fairly robust for larger models.

Dataset	n	s	w	τ	α
Synthetic HQMM	2	6	1	0.75	0.92
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
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

Table 12.1: **Hyperparameter Selection** The best performing step sizes (τ) and decay rates (α) for various ROSM 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.

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., 2017b) to quickly sample through many hyperparameter 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 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 trained 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 12.2, 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 is done in Python.

12.3 *Estimating Speedup*

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

by ROSM 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 ROSM and GS on the synthetic HMM data until one of them converges within a tolerance of 10^{-5} in DA scores. Since ROSM 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 ROSM. Note that a linear fit is an optimistic assumption of GS convergence time, meaning we are going to *understate* how much faster ROSM is compared to GS. Finally, we estimate the speed up offered by ROSM as the ratio of the (estimated) convergence time for GS and the actual convergence time for ROSM. In Figure 12.3, 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, ROSM offers comparable increase in speed up as parameters grow either by virtue of increasing n or w.

12.4 *Alternative Optimizations on the Stiefel Manifold*

In Figure 12.4, we compare learning HQMMs via various update schemes to constraint learned parameters on the Stiefel manifold on the synthetic HMM and synthetic HQMM datasets. All approaches yielded fairly similar results, with the Wen-Yin updates being slightly faster, especially for larger models on the synthetic HQMM data.

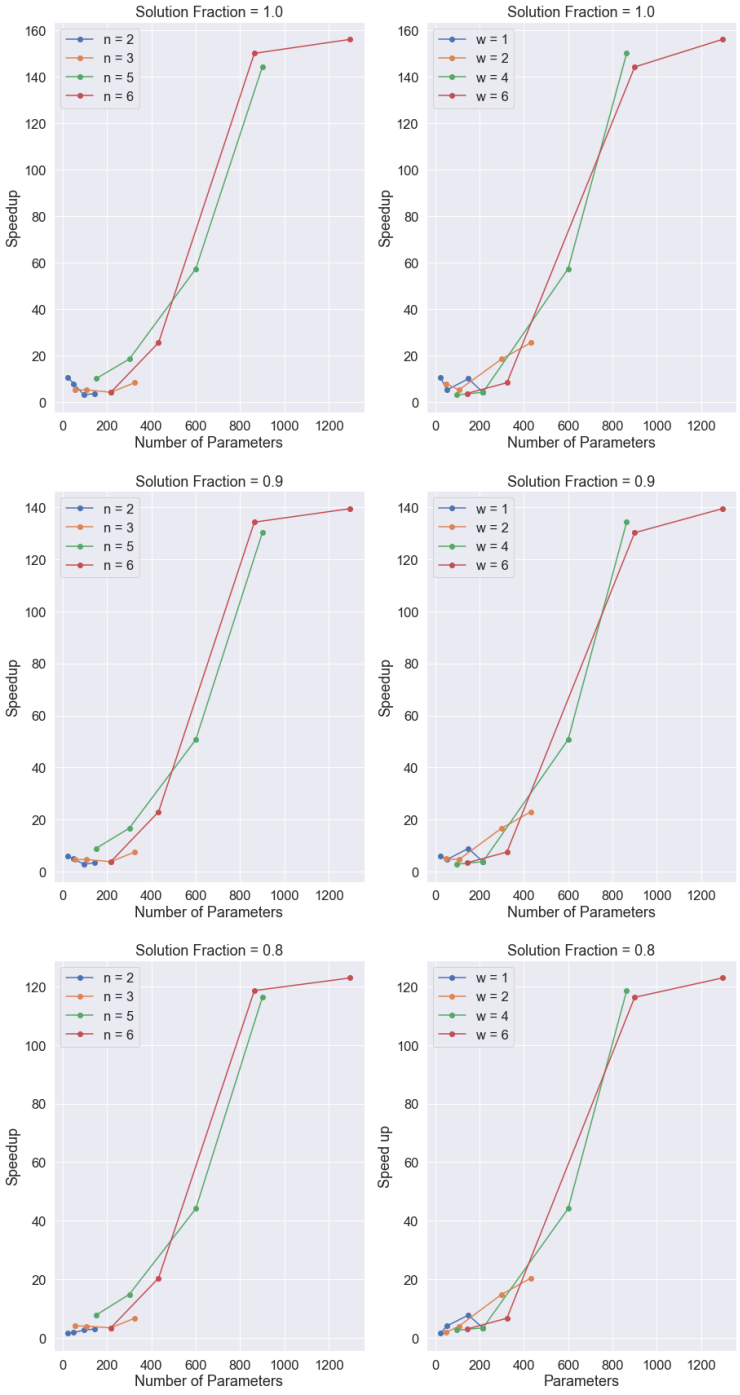
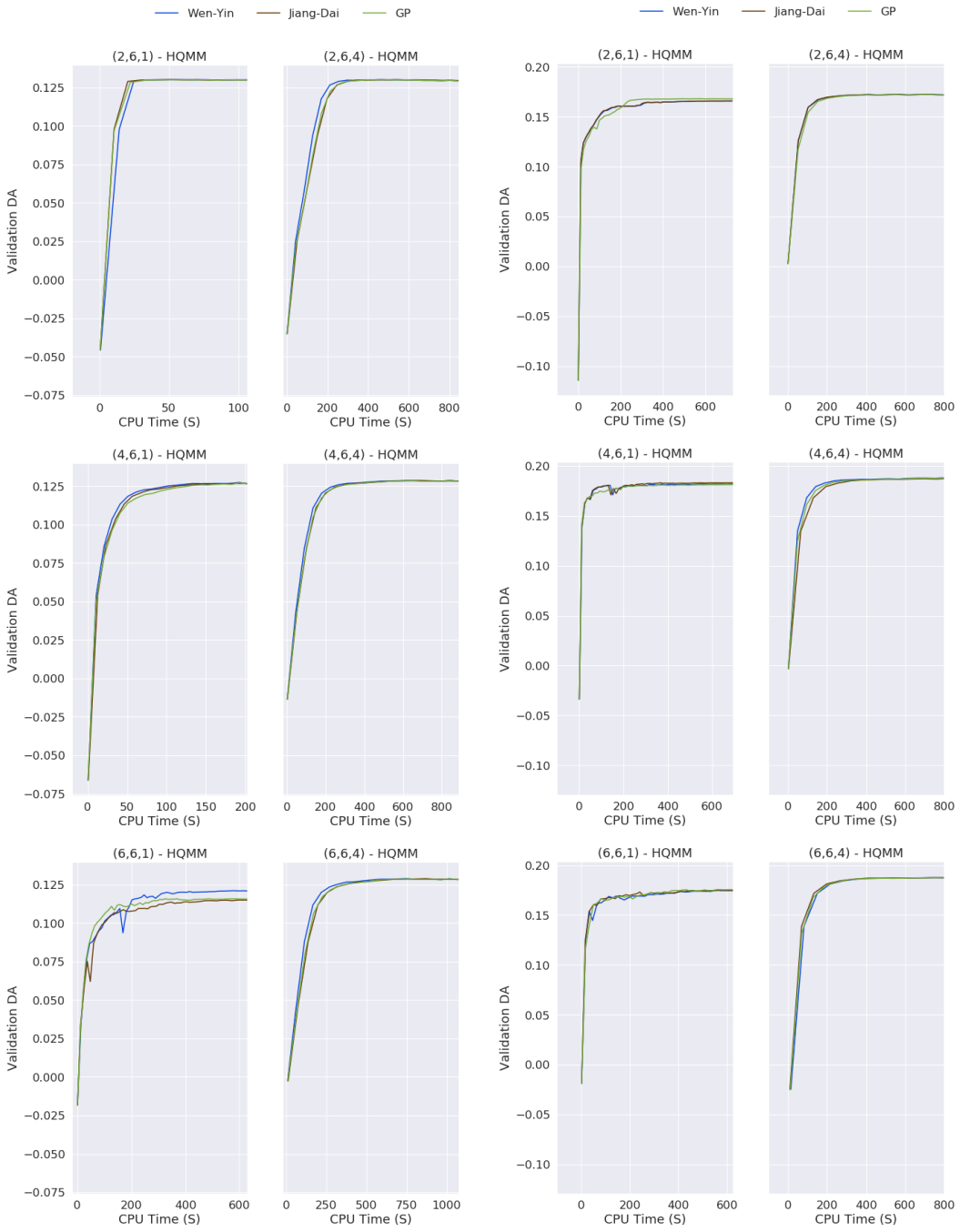


Figure 12.2: **Estimated Speedup of ROSM over GS**: Estimated speedups of ROSM over GS for various solution fractions. As seen in the plots for solution fraction of 1, GS can take more than 150 times the convergence time for ROSM to reach the latter's final solution quality.



(a) (b)

Figure 12.3: **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 accuracy, but the Wen-Yin update outperforms the others by a small margin.

13
Appendix: Metric Informed Sampling

13.1 Experiment Hyperparameters

Manifold	Model	Hyperparameter	Search Range	Selected Hyperparameter
S^3	Riemannian kernel herding	learning rate	1e-4 to 1e-2	0.1
		bandwidth	1e-4 to 100	10
	Euclidean kernel herding	learning rate	1e-4 to 1e-2	0.1
		bandwidth	1e-4 to 100	1e-4
	OT	learning rate	1e-4 to 1e-2	0.1
		regularization	1e-4 to 100	0.1
$SO(3)$	Riemannian kernel herding	learning rate	1e-4 to 1e-2	0.1
		bandwidth	1e-4 to 100	5
	Euclidean kernel herding	learning rate	1e-4 to 1e-2	0.001
		bandwidth	1e-4 to 100	1.0
	OT	learning rate	1e-4 to 1e-2	0.1
		regularization	1e-4 to 100	0.1
	OT + Cayley Centroid	regularization	1e-4 to 100	0.05

Table 13.1: **Hyperparameter Selection for Resampling Experiments**: Hyperparameters used for resampling experiments in Section 8.4.2

Experiment	Model	Hyperparameter	Search Range	Selected Hyperparameter
Gaussian Covariance Estimation	Riemannian kernel herding	learning rate	1e-4 to 1e-2	0.1
		param. bandwidth	1e-3 to 10 + median trick	0.01
		obs. bandwidth	1e-3 to 10 + median trick	0.1
		regularization	1e-9 to 10	1e-9
	Euclidean kernel herding	learning rate	1e-4 to 1e-2	1e-3
		param. bandwidth	1e-3 to 10 + median trick	median trick
		obs. bandwidth	1e-3 to 10 + median trick	0.1
		regularization	1e-9 to 10	1.0
	Euclidean kernel herding + Cholesky	learning rate	1e-4 to 1e-2	1e-3
		param. bandwidth	1e-3 to 10 + median trick	median trick
		obs. bandwidth	1e-3 to 10 + median trick	0.1
		regularization	1e-9 to 10	1.0
Inertia Matrix Estimation	Riemannian kernel herding	learning rate	1e-4 to 1e-2	0.1
		param. bandwidth	1e-3 to 10 + median trick	0.001
		obs. bandwidth	1e-3 to 10 + median trick	median trick
		regularization	1e-9 to 10	1e-9
	Euclidean kernel herding	learning rate	1e-4 to 1e-2	0.1
		param. bandwidth	1e-3 to 10 + median trick	10
		obs. bandwidth	1e-3 to 10 + median trick	median trick
		regularization	1e-9 to 10	1e-6
	Euclidean kernel herding + Cholesky	learning rate	1e-4 to 1e-2	0.01
		param. bandwidth	1e-3 to 10 + median trick	1.0
		obs. bandwidth	1e-3 to 10 + median trick	median trick
		regularization	1e-9 to 10	1e-9

Table 13.2: **Hyperparameter Selection for Simulator Parameter Estimation Experiments**: Hyperparameters used for Simulator Parameter Estimation Experiments in Section 8.5.2

14
Appendix: Metric Informed Reinforcement Learning

14.1 Preventing Feature Map Collapse and Explosion

Normalization Constraints Prevent Feature Map Explosion The normalization constraints of BeigeMaps constrain the expected value of the squared L_2 norm of the feature maps to be equal to the feature-dimension. Since BeigeMaps naturally enforce this constraint, additional heuristic clipping is not required.

This follows from standard properties of the normalization constraint of Laplacian eigenmaps. This constraint is enforced by the NeuralEF algorithm used in BeigeMaps as $\mathbb{E}(\vec{\phi} \odot \vec{\phi}) = \vec{1}$, where $\vec{\phi}_x$ is the learned eigenmap for state x, $\vec{1}$ is the all-ones vector, and $\odot$ is the element-wise product. Let us now consider the expected value of $\mathbb{E}(\|\phi(x_i)\|_2^2)$ for a batch of B embeddings $\phi(x_i) \in \mathbb{R}^D$

$$\begin{aligned}
\mathbb{E}(\|\phi(x)\|_2^2) &= \frac{1}{B}\sum_x \|\phi(x_i)\|_2^2 \\
&= \frac{1}{B}\sum_x \sum_i \phi(x_i)[i]\phi(x_i)[i] \\
&= \sum_i^d \frac{1}{B}\sum_x \phi(x_i)\phi(x_i)[i] \\
&= \sum_i^d \left(\frac{1}{B}\sum_x \phi(x_i)\phi(x_i)\right)[i] \\
&= \sum_i^d \mathbb{E}(\phi(x_i) \odot \phi(x_i))[i] \\
&= \sum_i^d \vec{1}[i] \quad \text{from the constraint} \\
&= D
\end{aligned}$$

Hence, $\mathbb{E}(\|\phi(x)\|_2^2) = D$. Empirically, this normalization constraint is enforced by the $L-2$ batch normalization layer applied after the last layer of the encoder ϕ. We also confirm the above result experimen-

tally, as shown in Figure 9.3.

Orthogonality Constraints Prevent Feature Map Collapse Feature map collapse is prevented by the orthogonality constraint of Laplacian eigenmaps $\Phi\mathbb{D}\Phi = \mathbb{I}$, where $\Phi \in \mathbb{R}^{B\times D}$ is the matrix of feature maps for D states, where the i-th row is the feature map for the i-th state x_i. Here, $\mathbb{D}$ is the diagonal degree matrix consisting column sums of the kernel. Since $\mathbb{D}$ is a positive definite matrix, we can write the orthogonality constraint as

$$\Phi^T\mathbb{D}^{1/2}\mathbb{D}^{1/2}\Phi = \mathbb{I}$$
$$(\mathbb{D}^{1/2}\Phi)^T(\mathbb{D}^{1/2}\Phi) = \mathbb{I}$$

Thus, the matrix $\mathbb{D}^{1/2}\Phi$ has full-column rank. This implies that Φ has rank D by using the property that $\texttt{rank}(AB) = \texttt{rank}(A)$ if C is an $B \times D$ matrix of rank D. Since the rank of Φ (i.e. the dimension of the sub-space spanned by the feature map) is constrained to be D, Laplacian eigenmaps naturally prevent feature map collapse. Similar arguments discussing the dimensionality of Φ can also be found in Belkin and Niyogi (2003). The NeuralEF algorithm used in BeigeMaps applies this orthogonlization constraint as a penalty in the overall objective.

14.2 Deep Mind Control Environments

In Table 14.1, we provide the environment parameters for the Deep Mind Control suite environments used in our experiments. The choice of action repeats for various environments has been noted as important, but can vary across implementations in the literature. We use the action repeats that were shown to work well in Yarats et al. (2019), which are also listed in Table 14.1.

14.3 Model Hyperparameters

We use the same model architecture as in Zhang et al. (2020), and attempt to match their implementation and hyperparameters as closely as possible. In Table 14.2, we list all model hyperparameter choices – the only difference from Zhang et al. (2020) is that we use a smaller replay buffer size ($100,000$ instead of $1M$) due to computational constraints. To enforce the normalization constraints for BeigeMaps in Equation 9.9, we apply an L_2 normalization layer at the end of the encoder following Deng et al. (2022b).

Parameter	Value
Image Dimension	3 x 88 x 88
Stacked Frames	3
Observation Dimension	9 x 88 x 88
Episode Length	1000
Action Repeats	
Finger Spin	2
Walker Walk	2
Walker Stand	2
Cheetah Run	4
Cartpole Swingup	8
Cartpole Balance	8
Ball in Cup Catch	4

Table 14.1: **DMC environment Parameters** Parameters used for all Deep Mind control experiments

14.4 *Learning Curves*

In Figure 14.1, we present normalized returns obtained from all models during training. As in the main text, the returns (which are within $[0, 1000]$) are normalized to be within $[0, 1]$. The curves correspond to the mean normalized returns of models during training when evaluated on evaluation environments with 5 separate random seeds. The curved regions correspond to the standard deviation in these mean estimates.

14.5 *Kernel Normalization*

As discussed in the main text, the literature on spectral clustering and dimensionality tend to use a few variations on how the Laplacian matrix (or in our case, the kernel matrix) is normalized. While different, all normalizations are intimately related, and share similar interpretations with respect to locality preservation and clustering. We consider two of the most popular alternatives: the symmetric normalization $K_{\text{sym}} = \mathbb{D}^{-1/2} K \mathbb{D}^{-1/2}$ and the random-walk normalization $K = \mathbb{D}^{-1} K$, following the nomenclature from Von Luxburg (2007). The relationship between these normalization choices is described in Proposition 3 in Von Luxburg (2007), which we summarize here.

The Laplacian eigenmaps corroesponding to the locality preserving objective in Equation 9.8 correspond to bottom solutions of the *generalized* eigenvalue problem $L\phi = \lambda \mathbb{D}\phi$ (Belkin and Niyogi, 2003). If (λ, ϕ) is an eigenvalue-eigenvector pair for the genralized eigenvalue problem, it is also an eigenpairs of the random-walk Laplacian $L_{rw} = \mathbb{D}^{-1} K$, i.e.$L_{rw}\phi = \lambda\phi$. Moreover, if (λ, ϕ) is an eigenpair of L_{rw}, then $(\lambda, \mathbb{D}^{1/2}\phi)$ is an eigenpair of $L_{sym} = \mathbb{D}^{-1/2} K \mathbb{D}^{-1/2}$. Thus

Hyperparameter	Value
Replay buffer capacity	**100,000**
Batch Size	128
Discount γ	0.99
Critic learning rate	10^{-5}
Critic target update frequency	2
Critic Q-function soft-update rate τ_Q	0.005
Critic encoder soft-update rate τ_ϕ	0.005
Actor learning rate	10^{-5}
Actor update frequency	2
Actor log stddev bounds	[-5,2]
Decoder learning rate	10^{-5}
Temperature learning rate	10^{-4}
Temperature Adam's β_1	0.9
Init Temperature	0.1
Dynamics hidden dimension	512
Encoder	
Encoder feature dimension	50
Encoder learning rate	10^{-5}
Number of convolutional layers	4
Num filters per convolution	32
Kernel size	3 x 3
Stride	2 for first, 1 for rest
Actor	
Actor MLP num layers	2
Actor MLP hidden dimension	256
Critic	
Critic MLP layers	2
Critic MLP hidden dimension	256
Gaussian Dynamics Model	
MLP num layers	1
MLP hidden dimension	512
Reward Prediction Modell	
MLP num layers	1
MLP hidden dimension	512

Table 14.2: **Model Hyperparameters for Behavioral Distance based Algorithms** Hyperparameters and model architecture of models used in Deep Mind control experiments

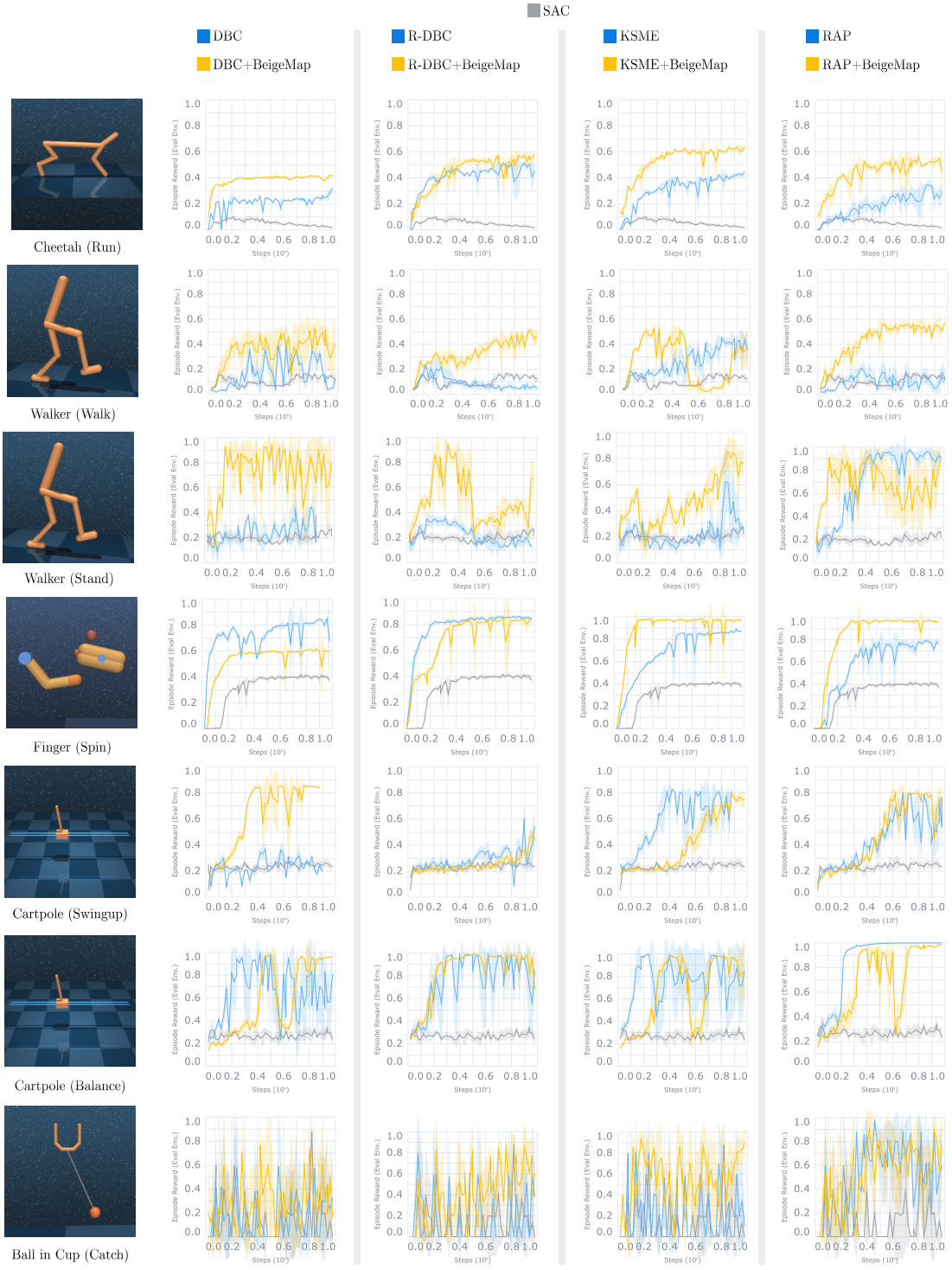


Figure 14.1: **Learning Curves for Deep Mind control experiments** Learning curves for all environments. Mean normalized returns on evaluation environment averaged over 5 random seeds.

the eigenspectrum of L_{rw} and L_{sym} only differ by a factor of $\mathbb{D}^{1/2}$. Note that the bottom eigenvectors of either normalized Laplacian corresponds to the top eigenvector of the corresponding normalized Kernel. When computing BeigeMaps, we use the normalized kernel instead of the normalized Laplacian.

In Figure 14.2, we compare two options for normalized kernels used in BeigeMaps: the symmetric (Sym) normalization $K = \mathbb{D}^{-1/2}K\mathbb{D}^{-1/2}$ and $K = \mathbb{D}^{-1}K$. In the main text, we present results for the symmetric normalization as it generally outperformed the random-walk normalization for all distances evaluated.

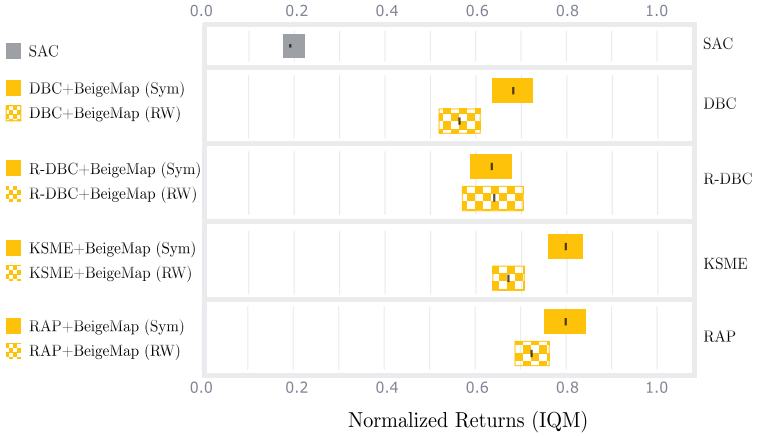


Figure 14.2: **Comparing Symmetric and Random Walk Normalizations for Kernels** Interquartile median of total normalized returns of BeigeMap algorithms using the symmetric (Sym) normalization $K = \mathbb{D}^{-1/2}K\mathbb{D}^{-1/2}$ and $K = \mathbb{D}^{-1}K$. The symmetric normalization outperformed the random walk variant for all behavioral distances, which motivates our choice of the symmetric normalization for our experiments.

Part VII

References

15
Bibliography

Absil, P.-A., Baker, C. G., and Gallivan, K. A. (2007a). Trust-region methods on riemannian manifolds. *Foundations of Computational Mathematics*, 7(3):303–330.

Absil, P.-A., Mahony, R., and Sepulchre, R. (2007b). *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, NJ, USA.

Adhikary, S. and Boots, B. (2022). Modular policy composition with policy centroids. In *2022 Multidisciplinary Conference on Reinforcement Learning and Decision Making (RLDM)*.

Adhikary, S., Li, A., and Boots, B. (2024). Beigemaps: Behavioral eigenmaps for reinforcement learning. In *Forty First International Conference on Machine Learning (International Conference on Machine Learning (ICML))*.

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

Adhikary, S., Srinivasan, S., Miller, J., Rabusseau, G., and Boots, B. (2021a). Quantum tensor networks, stochastic processes, and weighted automata. In *International Conference on Artificial Intelligence and Statistics*, pages 2080–2088. PMLR.

Adhikary, S., Srinivasan, S., Miller, J., Rabusseau, G., and Boots, B. (2021b). Quantum tensor networks, stochastic processes, and weighted automata. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 2080–2088. PMLR.

Agarwal, R., Machado, M. C., Castro, P. S., and Bellemare, M. G. (2021a). Contrastive behavioral similarity embeddings for generalization in reinforcement learning. *arXiv preprint arXiv:2101.05265*.

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A. C., and Bellemare, M. (2021b). Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320.

Altschuler, J., Niles-Weed, J., and Rigollet, P. (2017). Near-linear time approximation algorithms for optimal transport via sinkhorn iteration. *Advances in neural information processing systems*, 30.

Amari, S.-i. (1987). Differential geometrical theory of statistics. *Differential geometry in statistical inference*, 10:19–94.

Amari, S.-i. (2016). *Information geometry and its applications*, volume 194. Springer.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR.

Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404.

Arulampalam, M. S., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on signal processing*, 50(2):174–188.

Bach, F., Lacoste-Julien, S., and Obozinski, G. (2012). On the equivalence between herding and conditional gradient algorithms. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*.

Badia, A. P., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, Z. D., and Blundell, C. (2020). Agent57: Outperforming the atari human benchmark. In *International conference on machine learning*, pages 507–517. PMLR.

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

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

Baker, B., Akkaya, I., Zhokov, P., Huizinga, J., Tang, J., Ecoffet, A., Houghton, B., Sampedro, R., and Clune, J. (2022). Video pretraining (vpt): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, 35:24639–24654.

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

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

Barfoot, T. D. (2024). *State estimation for robotics*. Cambridge University Press.

Barp, A., Oates, C. J., Porcu, E., and Girolami, M. (2022). A riemannstein kernel method. *Bernoulli*, 28(4):2181–2208.

Barry, J., Barry, D. T., and Aaronson, S. (2014). Quantum partially observable markov decision processes. *Phys. Rev. A*, 90:032311.

Baum, L. E. and Eagon, J. A. (1967). An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology. *Bulleting of the American Mathematical Society*.

Baum, L. E. et al. (1972). An inequality and associated maximization technique in statistical estimation for probabilistic functions of markov processes. *Inequalities*, 3(1):1–8.

Baum, L. E. and Petrie, T. (1966). Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 37(6):1554–1563.

Bécigneul, G. and Ganea, O.-E. (2019). Riemannian adaptive optimization methods. In *International Conference on Machine Learning*.

Belkin, M. and Niyogi, P. (2001). Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in neural information processing systems*, 14.

Belkin, M. and Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396.

Belkin, M. and Niyogi, P. (2005). Towards a theoretical foundation for laplacian-based manifold methods. In *Journal of computer and system sciences (Print)*.

Bellemare, M. G., Dabney, W., Dadashi, R., Taïga, A. A., Castro, P. S., Roux, N. L., Schuurmans, D., Lattimore, T., and Lyle, C. (2019). A geometric perspective on optimal representations for reinforcement learning.

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

Bingham, C. (1974). An antipodally symmetric distribution on the sphere. *The Annals of Statistics*, pages 1201–1225.

Birdal, T., Arbel, M., Simsekli, U., and Guibas, L. J. (2020). Synchronizing probability measures on rotations via optimal transport. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1566–1576.

Bonneel, N. and Digne, J. (2023). A survey of optimal transport for computer graphics and computer vision. In *Computer Graphics Forum*, volume 42, pages 439–460. Wiley Online Library.

Borg, I. and Groenen, P. J. (2005). *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media.

Borgwardt, S. and Patterson, S. (2021). On the computational complexity of finding a sparse wasserstein barycenter. *Journal of Combinatorial Optimization*, 41(3):736–761.

Boumal, N. (2023). *An introduction to optimization on smooth manifolds*. Cambridge University Press.

Braun, M., Jaquier, N., Rozo, L., and Asfour, T. (2024). Riemannian flow matching policy for robot motion learning. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5144–5151.

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. (2021). Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*.

Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE journal on robotics and automation*, 2(1):14–23.

Brubaker, M., Salzmann, M., and Urtasun, R. (2012). A family of mcmc methods on implicitly defined manifolds. In *Artificial intelligence and statistics*, pages 161–172. PMLR.

Bullen, P. S. (2013). *Handbook of means and their inequalities*, volume 560. Springer Science & Business Media.

Byrne, S. and Girolami, M. (2013). Geodesic monte carlo on embedded manifolds. *Scandinavian Journal of Statistics*, 40(4):825–845.

Calinon, S. (2020). Gaussians on riemannian manifolds: Applications for robot learning and adaptive control. *IEEE Robotics & Automation Magazine*, 27(2):33–45.

Carlyle, J. W. and Paz, A. (1971). Realizations by stochastic finite automata. *Journal of Computer and System Sciences*, 5(1):26–40.

Castro, P. and Precup, D. (2010). Using bisimulation for policy transfer in mdps. In *Proceedings of the AAAI conference on artificial intelligence*, volume 24, pages 1065–1070.

Castro, P. S. (2020). Scalable methods for computing state similarity in deterministic markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10069–10076.

Castro, P. S., Kastner, T., Panangaden, P., and Rowland, M. (2021). Mico: Improved representations via sampling-based state similarity for markov decision processes. In *Neural Information Processing Systems*.

Castro, P. S., Kastner, T., Panangaden, P., and Rowland, M. (2023). A kernel perspective on behavioural metrics for markov decision processes. *Transactions on Machine Learning Research*.

Chan, G., Keselman, A., Nakatani, N., Li, Z., and White, S. (2016). Matrix product operators, matrix product states, and ab initio density matrix renormalization group algorithms. *The Journal of chemical physics*, 145 1:014102.

Chen, J. and Pan, S. (2022). Learning representations via a robust behavioral metric for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 35:36654–36666.

Chen, S., Zheng, L., and Wu, H. (2023). Riemannian representation learning for multi-source domain adaptation. *Pattern Recognition*, 137:109271.

Chen, W. Y., Barp, A., Briol, F.-X., Gorham, J., Girolami, M., Mackey, L., and Oates, C. (2019). Stein point markov chain monte carlo. In *International Conference on Machine Learning*, pages 1011–1021. PMLR.

Chen, W. Y., Mackey, L., Gorham, J., Briol, F.-X., and Oates, C. (2018). Stein points. In *International Conference on Machine Learning*.

Chen, Y., Welling, M., and Smola, A. (2010). Super-samples from kernel herding. In *The Conference on Uncertainty in Artificial Intelligence (UAI)*.

Cheng, C.-A., Mukadam, M., Issac, J., Birchfield, S., Fox, D., Boots, B., and Ratliff, N. (2021). Rmpflow: A geometric framework for generation of multitask motion policies. *IEEE Transactions on Automation Science and Engineering*, 18(3):968–987.

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

Chwialkowski, K., Strathmann, H., and Gretton, A. (2016). A kernel test of goodness of fit. In *International conference on machine learning*, pages 2606–2615. PMLR.

Cidre, G. A. (2016). Planning in a quantum system. *Masters Thesis, Carnegie Mellon University*.

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

Cohen, S. B., Stratos, K., Collins, M., Foster, D. P., and Ungar, L. (2013). 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.

Cohen, T. and Welling, M. (2016). Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR.

Cohen, T. S., Geiger, M., and Weiler, M. (2019). A general theory of equivariant cnns on homogeneous spaces. *Advances in neural information processing systems*, 32.

Comanici, G. and Precup, D. (2011). Basis function discovery using spectral clustering and bisimulation metrics. In *International Workshop on Adaptive and Learning Agents*, pages 85–99. Springer.

Coumans, E. and Bai, Y. (2016). Pybullet, a python module for physics simulation for games, robotics and machine learning.

Courty, N., Flamary, R., Tuia, D., and Rakotomamonjy, A. (2016). Optimal transport for domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 39(9):1853–1865.

Cover, T. M. (1999). *Elements of information theory*. John Wiley & Sons.

Critch, A. and Morton, J. (2014). Algebraic geometry of matrix product states. *Symmetry Integrability and Geometry-methods and Applications*, 10:095.

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

Cuturi, M. (2013). Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*.

Cuturi, M. and Doucet, A. (2014). Fast computation of wasserstein barycenters. In *International conference on machine learning*.

Dai, Y.-H. and Yuan, Y. (1999). A nonlinear conjugate gradient method with a strong global convergence property. *SIAM Journal on optimization*, 10(1):177–182.

Dayan, P. (1993). Improving generalization for temporal difference learning: The successor representation. *Neural computation*, 5(4):613–624.

De Bortoli, V., Mathieu, E., Hutchinson, M., Thornton, J., Teh, Y. W., and Doucet, A. (2022). Riemannian score-based generative modelling. *Advances in neural information processing systems*, 35:2406–2422.

Deng, Z., Shi, J., Zhang, H., Cui, P., Lu, C., and Zhu, J. (2022a). Neural eigenfunctions are structured representation learners. *ArXiv*, abs/2210.12637.

Deng, Z., Shi, J., and Zhu, J. (2022b). Neuralef: Deconstructing kernels by deep neural networks. In *International Conference on Machine Learning*, pages 4976–4992. PMLR.

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

Deshpande, I., Zhang, Z., and Schwing, A. G. (2018). Generative modeling using the sliced wasserstein distance. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3483–3491.

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

Diaconis, P., Holmes, S., Shahshahani, M., et al. (2013). Sampling from a manifold. In *Advances in modern statistical theory and applications: a Festschrift in honor of Morris L. Eaton*, pages 102–125. Institute of Mathematical Statistics.

Doucet, A., Johansen, A. M., et al. (2009). A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704):3.

Dreisigmeyer, D. W. (2007). Direct search algorithms over riemannian manifolds. *Submitted to SIOPT*.

Duan, Y., Ke, T., and Wang, M. (2019). State aggregation learning from markov transition data. *Advances in Neural Information Processing Systems*, 32.

Duncan, A., Nüsken, N., and Szpruch, L. (2023). On the geometry of stein variational gradient descent. *Journal of Machine Learning Research*, 24(56):1–39.

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

Dvurechensky, P., Gasnikov, A., and Kroshnin, A. (2018). Computational optimal transport: Complexity by accelerated gradient descent is better than by sinkhorn's algorithm. In *International conference on machine learning*, pages 1367–1376. PMLR.

Efroni, Y., Misra, D., Krishnamurthy, A., Agarwal, A., and Langford, J. (2021). Provably filtering exogenous distractors using multistep inverse dynamics. In *International Conference on Learning Representations*.

Eisert, J., Mueller, M., and Gogolin, C. (2012). Quantum measurement occurrence is undecidable. *Physical review letters*, 108 26:260501.

Ernst, J. and Kellis, M. (2012). Chromhmm: automating chromatin-state discovery and characterization. *Nature methods*, 9(3):215–216.

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

Fanizza, M., Lumbreras, J., and Winter, A. (2024). Quantum theory in finite dimension cannot explain every general process with finite memory. *Communications in Mathematical Physics*, 405(2):50.

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

Feragen, A. and Hauberg, S. (2016). Open problem: Kernel methods on manifolds and metric spaces. what is the probability of a positive definite geodesic exponential kernel? In *Conference on Learning Theory*.

Feragen, A., Lauze, F., and Hauberg, S. (2015). Geodesic exponential kernels: When curvature and linearity conflict. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Ferns, N., Castro, P. S., Precup, D., and Panangaden, P. (2006). Methods for computing state similarity in markov decision processes. *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence, UAI '06*.

Ferns, N., Panangaden, P., and Precup, D. (2004). Metrics for finite markov decision processes. In *UAI*, volume 4, pages 162–169.

Ferns, N., Panangaden, P., and Precup, D. (2011). Bisimulation metrics for continuous markov decision processes. *SIAM Journal on Computing*, 40(6):1662–1714.

Ferns, N., Panangaden, P., and Precup, D. (2012). Metrics for markov decision processes with infinite state spaces. *arXiv preprint arXiv:1207.1386*.

Ferns, N. and Precup, D. (2014). Bisimulation metrics are optimal value functions. In *UAI*, pages 210–219.

Feydy, J., Séjourné, T., Vialard, F.-X., Amari, S.-i., Trouvé, A., and Peyré, G. (2019). Interpolating between optimal transport and mmd using sinkhorn divergences. In *The 22nd international conference on artificial intelligence and statistics*, pages 2681–2690. PMLR.

Finzi, M., Stanton, S., Izmailov, P., and Wilson, A. G. (2020). Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In *International Conference on Machine Learning*, pages 3165–3176. PMLR.

Finzi, M., Welling, M., and Wilson, A. G. (2021). A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. In *International conference on machine learning*, pages 3318–3328. PMLR.

Fisher, R. A. (1953). Dispersion on a sphere. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 217(1130):295–305.

Flamary, R. and Courty, N. (2017). Pot python optimal transport library.

Fletcher, P., Lu, C., Pizer, S., and Joshi, S. (2004). Principal geodesic analysis for the study of nonlinear statistics of shape. *IEEE Transactions on Medical Imaging*, 23(8):995–1005.

Fliess, M. (1974). Matrices de hankel. *J. Math. Pures Appl*, 53(9):197–222.

Fu, X., Yang, G., Agrawal, P., and Jaakkola, T. (2021). Learning task informed abstractions. In *International Conference on Machine Learning*, pages 3480–3491. PMLR.

Fukumizu, K., Gretton, A., Sun, X., and Schölkopf, B. (2007). Kernel measures of conditional dependence. In *Conference on Neural Information Processing Systems*, volume 20.

Ganea, O., Bécigneul, G., and Hofmann, T. (2018). Hyperbolic neural networks. *Advances in neural information processing systems*, 31.

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

Garreau, D., Jitkrittum, W., and Kanagawa, M. (2017). Large sample analysis of the median heuristic. *arXiv: Statistics Theory*.

Gerlach, W. and Stern, O. (1922). Der experimentelle nachweis der richtungsquantelung im magnetfeld. *Zeitschrift für Physik*, 9(1):349–352.

Geyer, C. (2011). Introduction to markov chain monte carlo. *Handbook of markov chain monte carlo*, 20116022:45.

Ghosh, D. and Bellemare, M. G. (2020). Representations for stable off-policy reinforcement learning. In III, H. D. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3556–3565. PMLR.

Girolami, M. and Calderhead, B. (2011). Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214.

Givan, R., Dean, T., and Greig, M. (2003). Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 147(1-2):163–223.

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

Golub, G. H. and Loan, C. F. V. (1996). *Matrix Computations, Third Edition*. Johns Hopkins University Press.

Gorham, J. and Mackey, L. (2015). Measuring sample quality with stein's method. *Advances in neural information processing systems*, 28.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773.

Greydanus, S., Dzamba, M., and Yosinski, J. (2019). Hamiltonian neural networks. *Advances in neural information processing systems*, 32.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein gans. *Advances in neural information processing systems*, 30.

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

Gupta, V., Anand, D., Paruchuri, P., and Kumar, A. (2021). Action selection for composable modular deep reinforcement learning. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pages 565–573.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR.

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. (2019a). Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. (2019b). Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR.

Hamm, J. and Lee, D. (2008a). Extended grassmann kernels for subspace-based learning. *Advances in neural information processing systems*, 21:601–608.

Hamm, J. and Lee, D. D. (2008b). Grassmann discriminant analysis: a unifying view on subspace-based learning. In *Proceedings of the 25th international conference on Machine learning*, pages 376–383.

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

Hayes, C. F., Rădulescu, R., Bargiacchi, E., Källström, J., Macfarlane, M., Reymond, M., Verstraeten, T., Zintgraf, L. M., Dazeley, R., Heintz, F., et al. (2021). A practical guide to multi-objective reinforcement learning and planning. *arXiv preprint arXiv:2103.09568*.

Hestenes, M. R. and Stiefel, E. (1952). *Methods of conjugate gradients for solving linear systems*, volume 49. NBS Washington, DC.

Higham, N. J. (1988). Computing a nearest symmetric positive semidefinite matrix. *Linear Algebra and Its Applications*.

Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800.

Hjørungnes, A. and Gesbert, D. (2007). Complex-valued matrix differentiation: Techniques and key results. *IEEE Transactions on Signal Processing*, 55:2740–2746.

Ho, N., Nguyen, X., Yurochkin, M., Bui, H. H., Huynh, V., and Phung, D. (2017). Multilevel clustering via wasserstein means. In *International conference on machine learning*, pages 1501–1509. PMLR.

Honeine, P. and Richard, C. (2010). The angular kernel in machine learning for hyperspectral data classification. In *2010 2nd Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing*, pages 1–4. IEEE.

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

Hsu, K. and Ramos, F. (2019). Bayesian learning of conditional kernel mean embeddings for automatic likelihood-free inference. In *The 22nd International Conference on Artificial Intelligence and Statistics*.

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

Humphrys, M. (1995). W-learning: competition among selfish Q-learners. Technical Report UCAM-CL-TR-362, University of Cambridge, Computer Laboratory.

Hutsebaut-Buysse, M., Mets, K., and Latré, S. (2022). Hierarchical reinforcement learning: A survey and open research challenges. *Machine Learning and Knowledge Extraction*, 4(1):172–221.

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

Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., and Kavukcuoglu, K. (2016). Reinforcement learning with unsupervised auxiliary tasks. In *International Conference on Learning Representations*.

Jaeger, H. (1998). Discrete-time, discrete-valued observable operator models: A tutorial. Technical report.

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

Jaquier, N., Borovitskiy, V., Smolensky, A., Terenin, A., Asfour, T., and Rozo, L. (2022). Geometry-aware bayesian optimization in robotics using riemannian matérn kernels. In *Conference on Robot Learning*, pages 794–805. PMLR.

Jaquier, N., Rozo, L., Calinon, S., and Bürger, M. (2020). Bayesian optimization meets riemannian manifolds in robot learning. In *Conference on Robot Learning*.

Jayasumana, S., Hartley, R., Salzmann, M., Li, H., and Harandi, M. (2013a). Combining multiple manifold-valued descriptors for improved object recognition. In *2013 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–6. IEEE.

Jayasumana, S., Hartley, R., Salzmann, M., Li, H., and Harandi, M. (2013b). Kernel methods on the riemannian manifold of symmetric positive definite matrices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 73–80.

Jayasumana, S., Hartley, R., Salzmann, M., Li, H., and Harandi, M. (2015). Kernel methods on riemannian manifolds with gaussian rbf kernels. In *IEEE Transactions on Pattern Analysis and Machine Intelligenc*, pages 2464–2477.

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

Johannes, M. and Polson, N. (2010). Mcmc methods for continuous-time financial econometrics. In *Handbook of financial econometrics: Applications*, pages 1–72. Elsevier.

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

Kajihara, T., Kanagawa, M., Yamazaki, K., and Fukumizu, K. (2018). Kernel recursive abc: Point estimation with intractable likelihood. In *International Conference on Machine Learning*.

Kakade, S. M. (2001). A natural policy gradient. *Advances in neural information processing systems*, 14.

Kakade, S. M. (2003). *On the sample complexity of reinforcement learning*. University of London, University College London (United Kingdom).

Kanagawa, M., Nishiyama, Y., Gretton, A., and Fukumizu, K. (2016). Filtering with state-observation examples via kernel monte carlo filter. *Neural Comput.*, 28(2).

Kantorovich, L. V. (2006). On the translocation of masses. *Journal of mathematical sciences*, 133(4).

Karlsson, J. (1997). *Learning to solve multiple goals*. University of Rochester.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. (2021). Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440.

Kemertas, M. and Aumentado-Armstrong, T. (2021). Towards robust bisimulation metric learning. *Advances in Neural Information Processing Systems*, 34:4764–4777.

Kemertas, M. and Jepson, A. (2022). Approximate policy iteration with bisimulation metrics. *Transactions on Machine Learning Research*.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kisamori, K., Kanagawa, M., and Yamazaki, K. (2020). Simulator calibration under covariate shift with kernels. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

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

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

Kochurov, M., Karimov, R., and Kozlukov, S. (2020). Geoopt: Riemannian optimization in pytorch. *arXiv preprint arXiv:2005.02819*.

Koren, Y., Borenstein, J., et al. (2000). Potential field methods and inherent limitations for mobile robot navigation. In *International Conference on Robotics and Automation (ICRA)*.

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

Kruskal, J. B. (1964). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27.

Lacoste-Julien, S., Lindsten, F., and Bach, F. (2015). Sequential kernel herding: Frank-wolfe optimization for particle filtering. In *Artificial Intelligence and Statistics*.

Lamb, A., Islam, R., Efroni, Y., Didolkar, A. R., Misra, D., Foster, D. J., Molu, L. P., Chari, R., Krishnamurthy, A., and Langford, J. (2022). Guaranteed discovery of control-endogenous latent states with multi-step inverse models. *Transactions on Machine Learning Research*.

Lambert, M., Chewi, S., Bach, F., Bonnabel, S., and Rigollet, P. (2022). Variational inference via wasserstein gradient flows. *Advances in Neural Information Processing Systems*, 35:14434–14447.

Lange, S. and Riedmiller, M. (2010). Deep auto-encoder neural networks in reinforcement learning. In *The 2010 international joint conference on neural networks (IJCNN)*, pages 1–8. IEEE.

Lange, S., Riedmiller, M., and Voigtländer, A. (2012). Autonomous reinforcement learning on raw visual input data in a real world application. In *The 2012 international joint conference on neural networks (IJCNN)*, pages 1–8. IEEE.

Laroche, R., Fatemi, M., Romoff, J., and van Seijen, H. (2017). Multi-advisor reinforcement learning. *arXiv:1704.00756*.

Laroche, R. and Feraud, R. (2017). Reinforcement learning algorithm selection. *arXiv preprint arXiv:1701.08810*.

Laskin, M., Lee, K., Stooke, A., Pinto, L., Abbeel, P., and Srinivas, A. (2020a). Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33:19884–19895.

Laskin, M., Srinivas, A., and Abbeel, P. (2020b). Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pages 5639–5650. PMLR.

Lattimore, T., Hutter, M., and Sunehag, P. (2013). The sample-complexity of general reinforcement learning. In *International Conference on Machine Learning*, pages 28–36. PMLR.

Ledoit, O. and Wolf, M. (2003). Improved estimation of the covariance matrix of stock returns with an application to portfolio selection. *Journal of empirical finance*, 10(5):603–621.

Lee, A. X., Nagabandi, A., Abbeel, P., and Levine, S. (2020). Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *Advances in Neural Information Processing Systems*, 33:741–752.

Leifer, M. S. and Poulin, D. (2008). Quantum graphical models and belief propagation. *Annals of Physics*, 323(8):1899–1946.

Li, A., Cheng, C.-A., Rana, M. A., Xie, M., Van Wyk, K., Ratliff, N., and Boots, B. (2021). RMP^2: A Structured Composable Policy Class for Robot Learning. In *Robotics: Science and Systems (R:SS)*.

Li, C.-L., Chang, W.-C., Cheng, Y., Yang, Y., and Póczos, B. (2017a). Mmd gan: Towards deeper understanding of moment matching network. *Advances in neural information processing systems*, 30.

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

Li, N. and Stephens, M. (2003). Modeling linkage disequilibrium and identifying recombination hotspots using single-nucleotide polymorphism data. *Genetics*, 165(4):2213–2233.

Lin, L. J. (1993). Scaling up reinforcement learning for robot control. In *International Conference on Machine Learning (ICML)*.

Liu, C. and Zhu, J. (2018). Riemannian stein variational gradient descent for bayesian inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Liu, Q. (2017). Stein variational gradient descent as gradient flow. *Advances in neural information processing systems*, 30.

Liu, Q. and Wang, D. (2016). Stein variational gradient descent: a general purpose bayesian inference algorithm. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 2378–2386.

Liu, Q., Zhou, Q., Yang, R., and Wang, J. (2023). Robust representation learning by clustering with bisimulation metrics for visual reinforcement learning with distractions. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 8843–8851.

Liu, Y., Ramachandran, P., Liu, Q., and Peng, J. (2017). Stein variational policy gradient. *arXiv preprint arXiv:1704.02399*.

M. Zhao, H. J. (2007). Norm observable operator models. Technical report, Jacobs University.

Machado, M. C., Bellemare, M. G., and Bowling, M. (2017a). A laplacian framework for option discovery in reinforcement learning. In *International Conference on Machine Learning*, pages 2295–2304. PMLR.

Machado, M. C., Rosenbaum, C., Guo, X., Liu, M., Tesauro, G., and Campbell, M. (2017b). Eigenoption discovery through the deep successor representation. *In ICLR International Conference on Learning Representations*.

Mahadevan, S. (2005). Proto-value functions: Developmental reinforcement learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 553–560.

Mahadevan, S. and Maggioni, M. (2007). Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. *Journal of Machine Learning Research*, 8(10).

Makoviychuk, V., Wawrzyniak, L., Guo, Y., Lu, M., Storey, K., Macklin, M., Hoeller, D., Rudin, N., Allshire, A., Handa, A., et al. (2021). Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*.

Mardia, K. V. and Jupp, P. E. (2009). *Directional statistics*, volume 494. John Wiley & Sons.

Mardia, K. V., Taylor, C. C., and Subramaniam, G. K. (2007). Protein bioinformatics and mixtures of bivariate von mises distributions for angular data. *Biometrics*, 63(2):505–512.

Meilă, M. and Zhang, H. (2024). Manifold learning: What, how, and why. *Annual Review of Statistics and Its Application*, 11(1):393–417.

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

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

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

Milner, R. (1980). *A calculus of communicating systems*. Springer.

Miszczak, J. A. (2011). Singular value decomposition and matrix reorderings in quantum information theory. *International Journal of Modern Physics C*, 22(09):897–918.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

Moerland, T. M., Broekens, J., Plaat, A., Jonker, C. M., et al. (2023). Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 16(1):1–118.

Monge, G. (1781). Mémoire sur la théorie des déblais et des remblais. *Mem. Math. Phys. Acad. Royale Sci.*, pages 666–704.

Monras, A., Beige, A., and Wiesner, K. (2010). Hidden Quantum Markov Models and non-adaptive read-out of many-body states. *arXiv preprint arXiv:1002.2337*.

Montesuma, E. F. and Mboula, F. M. N. (2021). Wasserstein barycenter for multi-source domain adaptation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16785–16793.

Muandet, K., Balduzzi, D., and Schölkopf, B. (2013). Domain generalization via invariant feature representation. In *International conference on machine learning*, pages 10–18. PMLR.

Muandet, K., Fukumizu, K., Sriperumbudur, B., and Schölkopf, B. (2017). Kernel mean embedding of distributions: A review and beyond. *Found. Trends Mach. Learn.*, 10(1-2).

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

Nickel, M. and Kiela, D. (2017). Poincaré embeddings for learning hierarchical representations. *Advances in neural information processing systems*, 30.

Nielsen, F. (2020). An elementary introduction to information geometry. *Entropy*, 22(10):1100.

Nielsen, M. A. and Chuang, I. L. (2010). *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press.

Nielsen, M. A. and Chuang, I. L. (2011). *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, USA, 10th edition.

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

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

Nyström, E. J. (1930). Über die praktische auflösung von integralgleichungen mit anwendungen auf randwertaufgaben. *Acta Mathematica*, 54:185–204.

Oh, C., Gavves, E., and Welling, M. (2018). Bock: Bayesian optimization with cylindrical kernels. In *International Conference on Machine Learning*, pages 3868–3877. PMLR.

Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Orlin, J. (1988). A faster strongly polynomial minimum cost flow algorithm. In *Proceedings of the Twentieth annual ACM symposium on Theory of Computing*, pages 377–387.

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

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

Park, D. (1981). Concurrency and automata on infinite sequences. In *Theoretical Computer Science: 5th GI-Conference Karlsruhe, March 23–25, 1981*, pages 167–183. Springer.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR.

Pennec, X. (2006). Intrinsic statistics on riemannian manifolds: Basic tools for geometric measurements. *Journal of Mathematical Imaging and Vision*, 25:127–154.

Pennec, X., Fillard, P., and Ayache, N. (2006). A riemannian framework for tensor computing. *International Journal of computer vision*, 66:41–66.

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

Petrik, M. (2007). An analysis of laplacian methods for value function approximation in mdps. In *IJCAI*, pages 2574–2579.

Peyré, G., Cuturi, M., et al. (2019). Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607.

Pfau, D., Petersen, S., Agarwal, A., Barrett, D. G. T., and Stachenfeld, K. L. (2018). Spectral inference networks: Unifying deep and spectral learning. In *International Conference on Learning Representations*.

Pillis, J. (1967). Linear Transformations which Preserve Hermitian and Positive Semidefinite Operators. *Pacific Journal of Mathematics*.

Pineau, J., Gordon, G., and Thrun, S. (2003). Point-based value iteration: An anytime algorithm for pomdps. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, page 1025–1030, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Polyak, B. (1964). Some methods of speeding up the convergence of iteration methods. *Ussr Computational Mathematics and Mathematical Physics*, 4:1–17.

Qian, N. (1999a). On the momentum term in gradient descent learning algorithms. *Neural networks : the official journal of the International Neural Network Society*, 12 1:145–151.

Qian, N. (1999b). On the momentum term in gradient descent learning algorithms. *Neural Netw.*, 12(1):145–151.

Quiñonero-Candela, J., Sugiyama, M., Schwaighofer, A., and Lawrence, N. (2008). Covariate shift and local learning by distribution matching. *Dataset Shift in Machine Learning*, pages 131–160.

Qureshi, A. H., Johnson, J. J., Qin, Y., Henderson, T., Boots, B., and Yip, M. C. (2019). Composing task-agnostic policies with deep reinforcement learning. *arXiv preprint arXiv:1905.10681*.

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. In *Neural Information Processing Systems*.

Raissi, M., Perdikaris, P., and Karniadakis, G. (2019). Physics-informed neural networks: A deep learning framework for solving

forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707.

Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., and Levine, S. (2018). Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*.

Rawlings, J. B. (2000). Tutorial overview of model predictive control. *IEEE control systems magazine*, 20(3):38–52.

Reisinger, J., Waters, A., Silverthorn, B., and Mooney, R. J. (2010). Spherical topic models. In *International Conference on Machine Learning (ICML)*.

Ren, T., Zhang, T., Lee, L., Gonzalez, J. E., Schuurmans, D., and Dai, B. (2022). Spectral decomposition representation for reinforcement learning. *arXiv preprint arXiv:2208.09515*.

Rezaei-Shoshtari, S., Yurchyk, H., Fujimoto, S., Precup, D., and Meger, D. (2024). Fairness in reinforcement learning with bisimulation metrics. *arXiv preprint arXiv:2412.17123*.

Roijers, D. M., Vamplew, P., Whiteson, S., and Dazeley, R. (2013). A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48:67–113.

Rosenblatt, J. K. (1997). Damn: A distributed architecture for mobile navigation. *Journal of Experimental & Theoretical Artificial Intelligence*, 9(2-3):339–360.

Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323:533–536.

Russell, S. J. and Zimdars, A. (2003). Q-decomposition for reinforcement learning agents. In *Proceedings of the 20th International Conference on Machine Learning (International Conference on Machine Learning (ICML)-03)*, pages 656–663.

Saveriano, M., Abu-Dakka, F. J., and Kyrki, V. (2023). Learning stable robotic skills on riemannian manifolds. *Robotics and Autonomous Systems*, 169:104510.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2008). The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80.

Scherrer, B. (2013). Improved and generalized upper bounds on the complexity of policy iteration. *Advances in Neural Information Processing Systems*, 26.

Schölkopf, B. (2022). Causality for machine learning. In *Probabilistic and causal inference: The works of Judea Pearl*, pages 765–804.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Schützenberger, M. P. (1961). On the definition of a family of automata. *Inf. Control.*, 4(2-3):245–270.

Shaham, U., Stanton, K. P., Li, H., Nadler, B., Basri, R., and Kluger, Y. (2018). Spectralnet: Spectral clustering using deep neural networks. *ArXiv*, abs/1801.01587.

Shalit, U., Johansson, F. D., and Sontag, D. (2017). Estimating individual treatment effect: generalization bounds and algorithms. In *International conference on machine learning*, pages 3076–3085. PMLR.

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

Siddique, U., Weng, P., and Zimmer, M. (2020). Learning fair policies in multi-objective (deep) reinforcement learning with average and discounted rewards. In *International Conference on Machine Learning*, pages 8905–8915. PMLR.

Silver, D., Singh, S., Precup, D., and Sutton, R. S. (2021). Reward is enough. *Artificial intelligence*, 299:103535.

Simpkins, C. and Isbell, C. (2019). Composable modular reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4975–4982.

Singh, S., James, M. R., and Rudary, M. R. (2004). 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. AUAI Press.

Singh, S. P. (1992). The efficient learning of multiple task sequences. In *Advances in neural information processing systems*, pages 251–258.

Smith, S. T. (1994). Optimization techniques on riemannian manifolds. *Fields institute communications*, 3(3):113–135.

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

Smola, A. J. and Kondor, R. (2003). Kernels and regularization on graphs. In *Learning Theory and Kernel Machines: 16th Annual Conference on Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003. Proceedings*, pages 144–158. Springer.

Sommer, S., Lauze, F., Hauberg, S., and Nielsen, M. (2010). Manifold valued statistics, exact principal geodesic analysis and the effect of linear approximations. In Daniilidis, K., Maragos, P., and Paragios, N., editors, *Computer Vision – ECCV 2010*, pages 43–56, Berlin, Heidelberg. Springer Berlin Heidelberg.

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

Song, Y., Suganthan, P. N., Pedrycz, W., Ou, J., He, Y., Chen, Y., and Wu, Y. (2023). Ensemble reinforcement learning: A survey. *Applied Soft Computing*, 149:110975.

Sprague, N. and Ballard, D. (2003). Multiple-goal reinforcement learning with modular sarsa (0). *Arxiv*.

Sra, S. (2018). Directional statistics in machine learning: a brief review. *Applied Directional Statistics: Modern Methods and Case Studies*, 225.

Srinivasan, S., Downey, C., and Boots, B. (2018a). Learning and inference in hilbert space with quantum graphical models. In *Advances in Neural Information Processing Systems*, pages 10338–10347.

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

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

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

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

Stachenfeld, K. L., Botvinick, M., and Gershman, S. J. (2014). Design principles of the hippocampal cognitive map. *Advances in neural information processing systems*, 27.

Stinespring, W. F. (1955). Positive functions on c*-algebras. *Proceedings of the American Mathematical Society*, 6(2):211–216.

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

Stooke, A., Lee, K., Abbeel, P., and Laskin, M. (2021). Decoupling representation learning from reinforcement learning. In *International Conference on Machine Learning*, pages 9870–9879. PMLR.

Stoudenmire, E. and Schwab, D. J. (2016). Supervised learning with tensor networks. In *Advances in Neural Information Processing Systems*, pages 4799–4807.

Tang, H., Chu, S. M., and Huang, T. S. (2009). Generative model-based speaker clustering via mixture of von mises-fisher distributions. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*.

Tang, Y. and Agrawal, S. (2018). Implicit policy for reinforcement learning. *arXiv preprint arXiv:1806.06798*.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., de Las Casas, D., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., Lillicrap, T. P., and Riedmiller, M. A. (2018). Deepmind control suite. *ArXiv*, abs/1801.00690.

Taylor, J., Precup, D., and Panagaden, P. (2008). Bounding performance loss in approximate mdp homomorphisms. *Advances in Neural Information Processing Systems*, 21.

Tenenbaum, J. B., Silva, V. d., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323.

Tham, C. K. and Prager, R. W. (1994). A modular q-learning architecture for manipulator task decomposition. In *Machine Learning Proceedings 1994*, pages 309–317. Elsevier.

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

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics*. MIT Press, Cambridge, MA.

Tifrea, A., Bécigneul, G., and Ganea, O.-E. (2018). Poincaré glove: Hyperbolic word embeddings. *ArXiv*, abs/1810.06546.

Todorov, E., Erez, T., and Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE.

Tolstikhin, I., Bousquet, O., Gelly, S., and Schoelkopf, B. (2017). Wasserstein auto-encoders. *arXiv preprint arXiv:1711.01558*.

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

Townsend, J., Koep, N., and Weichwald, S. (2016). Pymanopt: A python toolbox for optimization on manifolds using automatic differentiation. *J. Mach. Learn. Res.*, 17(137).

Urain, J., Anqi, L., Puze, L., D'eramo, C., and Peters, J. (2021). Composable energy policies for reactive motion generation and reinforcement learning. In *Robotics: Science and Systems XVII (R:SS 2021)*.

Vamplew, P., Smith, B. J., Källström, J., Ramos, G., Rădulescu, R., Roijers, D. M., Hayes, C. F., Heintz, F., Mannion, P., Libin, P. J., et al. (2022). Scalar reward is not enough: A response to silver, singh, precup and sutton (2021). *Autonomous Agents and Multi-Agent Systems*, 36(2):41.

Van Hasselt, H., Doron, Y., Strub, F., Hessel, M., Sonnerat, N., and Modayil, J. (2018). Deep reinforcement learning and the deadly triad. *arXiv preprint arXiv:1812.02648*.

van Seijen, H., Fatemi, M., Laroche, R., Romoff, J., Barnes, T., and Tsang, J. (2017). Hybrid reward architecture for reinforcement learning. In *NIPS*.

van Seijen, H., Fatemi, M., Romoff, J., and Laroche, R. (2016). Separation of concerns in reinforcement learning. *arXiv preprint arXiv:1612.05159*.

Van Seijen, H., Fatemi, M., Romoff, J., Laroche, R., Barnes, T., and Tsang, J. (2017). Hybrid reward architecture for reinforcement learning. *arXiv preprint arXiv:1706.04208*.

Van Wyk, K., Xie, M., Li, A., Rana, M. A., Babich, B., Peele, B., Wan, Q., Akinola, I., Sundaralingam, B., Fox, D., et al. (2022). Geometric

fabrics: Generalizing classical mechanics to capture the physics of behavior. *IEEE Robotics and Automation Letters*, 7(2):3202–3209.

Vidyasagar, M. (2011). 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.

Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing*, 17:395–416.

Wang, K., Zhou, K., Zhang, Q., Shao, J., Hooi, B., and Feng, J. (2021). Towards better laplacian representation in reinforcement learning with generalized graph drawing. In *International Conference on Machine Learning*, pages 11003–11012. PMLR.

Wang, R., Cheng, Y., and Wang, X. (2025). Constrained visual representation learning with bisimulation metrics for safe reinforcement learning. *IEEE Transactions on Image Processing*.

Wang, T., Du, S., Torralba, A., Isola, P., Zhang, A., and Tian, Y. (2022). Denoised mdps: Learning world models better than the world itself. In *International Conference on Machine Learning*, pages 22591–22612. PMLR.

Wang, Z. and Solo, V. (2020). Lie group state estimation via optimal transport. In *ICASSP*.

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

White, S. R. (1992). Density matrix formulation for quantum renormalization groups. *Phys. Rev. Lett.*, 69:2863–2866.

Wiering, M. A. and Van Hasselt, H. (2008). Ensemble algorithms in reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(4):930–936.

Wiewiora, E. (2007). *Modeling probability distributions with predictive state representations*. PhD thesis, University of California, San Diego.

Williams, C. K. I. and Seeger, M. W. (2000). Using the nyström method to speed up kernel machines. In *Neural Information Processing Systems*.

Wood, C. J., Biamonte, J. D., and Cory, D. G. (2015a). Tensor networks and graphical calculus for open quantum systems. *Quantum Info. Comput.*, 15(9-10):759–811.

Wood, C. J., Biamonte, J. D., and Cory, D. G. (2015b). Tensor networks and graphical calculus for open quantum systems. *Quantum Inf. Comput.*, 15:759–811.

Wu, J.-L., Xiao, H., and Paterson, E. (2017). Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework. *Physical Review Fluids*.

Wu, Y., Tucker, G., and Nachum, O. (2018). The laplacian in rl: Learning representations with efficient approximations. *ArXiv*, abs/1810.04586.

Yarats, D., Fergus, R., and Kostrikov, I. (2021a). Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *9th International Conference on Learning Representations, ICLR 2021*.

Yarats, D., Fergus, R., Lazaric, A., and Pinto, L. (2021b). Reinforcement learning with prototypical representations. In *International Conference on Machine Learning*, pages 11920–11931. PMLR.

Yarats, D., Zhang, A., Kostrikov, I., Amos, B., Pineau, J., and Fergus, R. (2019). Improving sample efficiency in model-free reinforcement learning from images. In *AAAI Conference on Artificial Intelligence*.

Yeang, C.-H. (2010). A probabilistic graphical model of quantum systems. In *Machine Learning and Applications (International Conference on Machine Learning (ICML)), 2010 Ninth International Conference on*, pages 155–162. IEEE.

Yen-Chen, L., Zeng, A., Song, S., Isola, P., and Lin, T.-Y. (2020). Learning to see before learning to act: Visual pre-training for manipulation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7286–7293. IEEE.

Yilmaz, A. and Shah, M. (2008). A differential geometric approach to representing the human actions. *Computer Vision and Image Understanding*, 109(3):335–351.

Ying, S. and Ying, M. (2014). Reachability analysis of quantum markov decision processes. *Inf. Comput.*, 263:31–51.

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

Zeestraten, M. J., Havoutis, I., Silvério, J., Calinon, S., and Caldwell, D. G. (2017). An approach for imitation learning on riemannian manifolds. *IEEE Robotics and Automation Letters*, 2(3):1240–1247.

Zhang, A., McAllister, R., Calandra, R., Gal, Y., and Levine, S. (2020). Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*.

Zhang, T., Ren, T., Yang, M., Gonzalez, J., Schuurmans, D., and Dai, B. (2022). Making linear mdps practical via contrastive representation learning. In *International Conference on Machine Learning*, pages 26447–26466. PMLR.

Zhang, Z., Chen, Y., Lee, J. D., and Du, S. S. (2024). Settling the sample complexity of online reinforcement learning. In *The Thirty Seventh Annual Conference on Learning Theory*, pages 5213–5219. PMLR.

Zhao, M. and Jaeger, H. (2010a). Norm-observable operator models. *Neural Computation*, 22:1927–1959.

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

Zhu, X. (2014). The optimal control related to riemannian manifolds and the viscosity solutions to hamilton–jacobi–bellman equations. *Systems & Control Letters*, 69:7–15.

Życzkowski, K. and Bengtsson, I. (2004). On Duality between Quantum Maps and Quantum States. *Open Systems & Information Dynamics*, 11(1):3–42.