UNIVERSITY OF CALIFORNIA SAN DIEGO

Scalable Measure Transportation and Applications in Machine Learning and Human Computer Interfaces

A dissertation submitted in partial satisfaction of the
requirements for the degree of Doctor of Philosophy

in

Computer Science (Computer Engineering)

by

Justin Cua Tantiongloc

Committee in charge:

Professor Todd P. Coleman, Co-Chair
Professor William Griswold, Co-Chair
Professor Garrison Cottrell
Professor Sanjoy Dasgupta
Professor Terry Jernigan

2018

The Dissertation of Justin Cua Tantiongloc is approved and is acceptable in quality and form for publication on microfilm and electronically:

Co-Chair

Co-Chair

University of California San Diego

2018

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to acknowledge Professor Todd P. Coleman, who has been my advisor for almost seven full years. The completion of my program and all the work described in this dissertation would not have been possible without his guidance and support during my time at UCSD. Over all these years, he has always been there when I needed it, and has not only been a fantastic mentor, but has also become a great friend.

I would also like to acknowledge Professor William Griswold, who has served as my co-advisor since I entered the PhD program. Bill has been greatly supportive of my efforts throughout the years, ever since I took Software Engineering courses with him during my first 2 years at UCSD, and has been a fantastic co-advisor.

I would also very much like to acknowledge Professor Gary Cottrell, Professor Sanjoy Dasgupta, and Professor Terry Jernigan, for being on my graduate committee, and taking the time out of their teaching and research lives to hear and read about my work.

And finally, I would also like to acknowledge the entire Coleman Lab, many of whom I have formed extremely valuable collaborations with, both for work described in this dissertation, as well as other projects that gave me the experience and knowledge I needed to get to this point. Beyond that, I thank the Lab as well for the countless meetings, presentations, and chats that have both educated and inspired me throughout the years.

Chapters 2,3,4 and 5 feature the reprint of much of the text in “A distributed framework for the construction of transport maps”, Mesa, Diego A.; Tantiongloc, Justin; Mendoza, Marcela; Coleman, Todd P., which has been submitted to Neural Computation in 2018, [52]. The dissertation author was a primary co-investigator of this work.

Chapter 5 additionally contains applications originally described in “An information and control framework for optimizing user-compliant human computer interfaces”, Proceedings of the IEEE, Tantiongloc, Justin; Mesa, Diego; Ma, Rui; Kim, Sanggyun; Alzate, Cristian H; Camacho, Jaime J; Manian, Vidya; Coleman, Todd P., 2017, [73], of which the dissertation

author was the primary investigator, and “Diffeomorphism learning via relative entropy constrained optimal transport”, GlobalSIP 2016, Coleman, Todd P.; Tantiongloc, Justin; Allegra, Alexis; Mesa, Diego A.; Kang, Dae; Mendoza, Marcela, 2016, [16], of which the dissertation author was the primary investigator as well as presenter at the GlobalSIP conference in 2016. I would also like to acknowledge Dr. Vidya Manian, Cristian Alzate, and Jaime Camacho, of the University of Puerto Rico, Mayagüez, without whom the brain-computer interface application described in [73] and in this section would not have been possible.

VITA

2011	Bachelor of Science, University of California Davis
2014	Master of Science, University of California San Diego
2018	Doctor of Philosophy, University of California San Diego
2012-2018	Graduate Researcher, Neural Interaction Lab of Dr. Todd P. Coleman, University of California San Diego

PUBLICATIONS

"A distributed framework for the construction of transport maps", Diego A. Mesa, Justin Tantiongloc, Marcela Mendoza and Todd P. Coleman, *Submitted* to Neural Computation, 2018

"An information and control framework for optimizing user–compliant human–computer interfaces", Justin Tantiongloc, Diego A. Mesa, Rui Ma, Sanggyun Kim, Cristian H. Alzate, Jaime J. Camacho, Vidya Manian, and Todd P. Coleman, Proceedings of the IEEE 105, no. 2 (2017): 273-285

"Diffeomorphism learning via relative entropy constrained optimal transport", Todd P. Coleman, Justin Tantiongloc, Alexis Allegra, Diego Mesa, Dae Kang, and Marcela Mendoza, Signal and Information Processing (GlobalSIP), 2016 IEEE Global Conference on, pp. 1330-1334. IEEE, 2016

ABSTRACT OF THE DISSERTATION

Scalable Measure Transportation and Applications in Machine Learning and Human Computer Interfaces

by

Justin Cua Tantiongloc

Doctor of Philosophy in Computer Science (Computer Engineering)

University of California San Diego, 2018

Professor Todd P. Coleman, Co-Chair
Professor William Griswold, Co-Chair

The field of optimal transportation is a broad area of theory pertaining to the computation of a mapping to transform one probability distribution into another. Many computational strategies solving the transport problem spanning decades of research have led to several interesting applications in areas such as statistics, economics, machine learning and computer science. However, there also exist many limitations to these algorithms, including the ever-present difficulty of scaling to larger datasets, both with respect to dimension and number of available samples from which we would like to extract useful information. Furthermore,

although we have seen a dramatic increase in the availability of powerful distributed computational resources throughout the past few decades, such as publicly available CPU clusters and GPU compute nodes, many modern algorithmic approaches to the transport problem are not specifically designed with parallelism in mind to accommodate such frameworks. In a world where the continuous interaction between human users and distributed computational resources drives our technologically modern lives, this capability is critical, especially with applications that are expected to work in real-time.

Building upon previous research from our group, this work investigates a parallelized, computational problem to create optimal transport maps that is designed with the notion of remote scalability in mind; this work focuses on a CUDA-based implementation of the framework that utilizes GPU computational resources to accommodate analysis of data in higher dimensions not often seen in the field of computational optimal transportation, and that furthermore scale with the availability of additional hardware. We will also present applications using this framework in several different areas of machine learning with an emphasis on human-computer interfaces, including a novel multi-user brain-computer interface, a classification problem pertaining to automated sleep-staging based on electroencephalographical (EEG) data, and an example of generative modeling using the MNIST dataset. Finally, we'll discuss a future direction for application of this framework to preference-based deep reinforcement learning.

Introduction

Optimal transportation is a broad field of study involving the computation of a mapping function, otherwise known as a *transport map*, to transform one probability distribution into another. Over the span of several centuries, we have witnessed an extremely impressive evolution of research in this area, as well as seen applications of the framework to a wide variety of problem areas ranging from economics, to physics, and even to machine learning. While the high-level statement of the problem is seemingly a simple one, there is quite the large number of theoretical considerations underlying the problem that gives rise to the wealth of potential applications it can accommodate, and the understanding of these theoretical aspects of the problem reveal just how powerful of an approach optimal transportation can be.

Building upon previous work from our group [48, 41, 51], the primary goal of this dissertation is to showcase a recent advance in the optimal transportation literature achieved by our research group and myself, in the form of a novel distributed, computationally efficient optimization problem and algorithm to compute such optimal transportation mappings. We very much stand on the shoulders of giants, and build upon a wealth of previous literature to theoretically justify and solidify our approach as a viable alternative, and at times, even preferred methodology of performing optimal transportation relative to the many computational approaches presented by those who have come before us.

In addition to deriving our optimization problem to compute a transport map, we also investigate the notion of sequential mappings where several individual maps are chained together to achieve an overall desired transformation and evaluate the advantages and disadvantages

to this approach. Furthermore, we will discuss the details and performance of a CUDA-based parallel implementation in which we have instantiated our problem. Finally, we will highlight several interesting applications of our framework in the realm of machine learning, with an emphasis on human-computer interfaces. In doing this, our aim is to showcase how this approach can become yet another powerful instrument in the machine learning scientist's toolbox.

This dissertation will be organized as follows: Chapter 1 will begin with a brief walk through the history of optimal transportation, and tell the story of how the problem formulation began and later evolved over several centuries of research and investigation. We will also present several of the most important theoretical conclusions rooted in previous literature that our approach relies on in showing feasibility and correctness. Thus, this chapter will serve as a critical historic foundation of all theoretical discussion in subsequent chapters.

Chapter 2 will revisit our group's previous work on optimal transport using a Kullback-Leibler (KL) divergence-based optimization problem. This will provide the theoretical foundation from which the novel work within this dissertation will be built.

Chapter 3 begins by examining two of the important limitations of the work in Chapter 2: scalability and the potential need for regularization. To address the issue of scalability, we will further revisit previous work from our group regarding the formulation of a parallelizable version of the KL-based formulation from Chapter 2 via the Alternating Directions Method of Multipliers (ADMM) [51]. After describing the theoretical aspects of the original problem, we will dive into the novel investigation contained within this dissertation, and identify several areas where the algorithm can be further modified and improved to the end of accommodating a wider variety of problems. This analysis culminates in the formulation of a slightly different optimization problem relative to [51] that specifically takes advantage of the Knothe-Rosenblatt rearrangement to represent our mappings in a more computationally feasible manner. Additionally, we will examine the scenario where several maps are chained together to form a sequence of transport mappings that are intended to achieve a desired overall transformation, and develop

some theoretical underpinnings explaining why this approach makes sense. In this context, we present a modified version of the original objective function that has an additional Wasserstein distance-based loss term associated with it, and discuss how this term affects the behavioral aspects of the sequential mapping, and why it can be useful.

Chapter 4 will primarily serve as an implementation-based discussion of the theoretical framework described in Chapter 3. We will discuss methods through which we instantiated all the theory from previous sections into actionable code design that ultimately made it into our current CUDA implementation of the optimal transport framework. We will also discuss concrete ways to make certain aspects of the theoretical implementation convenient to implement, and less abstract than the more theory-oriented discussion in previous sections. Finally, we present a few metrics and benchmarks for evaluating how the implementation performs on either a single GPU or multi-GPU system.

Chapter 5 will purely be dedicated to discussing several different applications of the framework to real-world example problems. The first application will be a reproduction of work from [73], where we apply our optimal transport framework to the theory of Posterior Matching [70] and construct a multi-user brain-computer interface. Next, we will instantiate our framework in the context of classification of electroenchephalography (EEG) signals for automated sleep stage identification based on results published in [16]. We then demonstrate the relatively high-dimensional optimal transportation capabilities of our framework via generative modeling of handwritten digits from the MNIST dataset, as originally reported in [52]. And finally, we also introduce a proposed future direction and proof-of-concept demonstration of an application of our optimal transportation methodology to the field of deep reinforcement learning, emphasizing the computational scalability of the algorithm and how it may potentially be applicable to situations with near-real-time constraints.

Chapter 1

A Background on Optimal Transportation

Here, we provide a high level discussion about the nature of the optimal transportation problem and a brief history of optimal transportation research.

1.1 The Monge and Kantorovich Formulations

The purpose of optimal transportation is to identify a mapping function to transform one probability density into another, while minimizing some measure of transport cost. One intuitive, physical example we can use to envision this idea is to imagine the process of reshaping a pile of sand from one arrangement into another, while trying to minimize the average distance that each individual grain of sand must travel to achieve the rearrangement. See Figure 1.1 for a simple, graphical representation of this process.

Within the optimal transport literature, there are two texts that offer very comprehensive analyses and treatments of the subject: *Optimal Transport: Old and New* by Cédric Villani [76], and *Optimal Transport for Applied Mathematicians* by Filippo Santambrogio [65]. The overall scope of both of these texts is well beyond the work that will be discussed in this thesis, but in this section we will discuss the basic background of optimal transport that will be necessary for the analyses in subsequent chapters. For an in-depth understanding of the mathematical theory of optimal transport however, there is really no substitute for reading these two, very detailed,

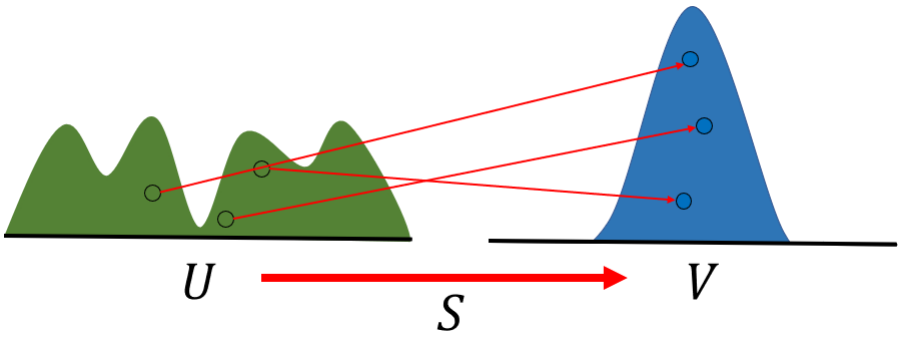


Figure 1.1. In optimal transportation, we seek a mapping S that transforms a source, or "push-from" distribution, U, to a target, or "push-to" distribution, V, while minimizing a transport cost, such as the average Euclidean distance between moving a mass from a location in U to a location in V.

texts.

More formally, in optimal transportation, we seek a map S that is defined as follows:

Definition 1.1 (Push-Forward) *Let* W *be a convex subset of D-dimensional Euclidean space,* $\mathsf{P}(\mathsf{W})$ *be the set of all probability measures on* W*, and* $U, V \in \mathsf{P}(\mathsf{W})$ *be probability measures. A transport map S transforms, or "pushes", U to V (denoted as* $S\#U = V$*), if a random variable X with distribution U results in* $Y \triangleq S(X)$ *having distribution V.*

The earliest form of the full optimal transportation problem applied to densities was formalized in 1781 by Gaspard Monge, and with the definition above, we can express the general optimal transportation problem as:

$$\inf_{S:\mathsf{W}\to\mathsf{W}} \int_{w\in\mathsf{W}} c(w, S(w)) dU(w) \tag{1.1}$$
$$\text{s.t.} \quad S\#U = V$$

where $c(x,y) : \mathsf{W} \times \mathsf{W} \to \mathbb{R}_+$ is some cost function associated with transporting x to y, and for which the optimal minimizer of (1.1) will be denoted as S^*.

Leonid Kantorovich revisited the optimal transport problem in 1942, and derived a relaxation to Monge's original formulation. In the Kantorivich problem, the aim is to identify a coupling between U and V, also called a *transport plan*, that we denote as $\gamma : \mathsf{W} \times \mathsf{W} \to \mathbb{R}$. The transport plan can be interpreted in several ways, but perhaps the most effective is to consider it is in the following way: if γ is an optimal transport plan from U to V, $\gamma(x,y)$ denotes how much mass out of u is being transferred from a coordinate x to a coordinate y in v. In other words, Kantorovich's formulation is a relaxation of Monge's problem in the sense that it allows for mass from a single source to be split among several destinations during transport from U to V. Kantorovich's problem can be written as:

$$\inf_{\gamma \in \Gamma(u,v)} \int_{\mathsf{W}\times\mathsf{W}} c(x,y) d\gamma(x,y) \tag{1.2}$$

where $\Gamma(u,v)$ represents the set of all measures on the product space $\mathsf{W} \times \mathsf{W}$ that have marginals equal to u and v. In other words, Kantorovich's problem relies on this feasible set to impose the $S\#U = V$ constraint. In general, (1.2) will have a smaller value than (1.1) by virtue of the fact that (1.1) always induces a coupling, but the coupling may have to split probability masses during the transport (e.g. in the case of discrete probability distributions).

Furthermore, when the cost function c is of the form $d(x,y)^p$ for some integer p, the cost of the transport using an optimal transport map is known as the *p-Wasserstein Distance*, and is a distance metric on the space of probability distributions [13, 76]. We can define the p-Wasserstein Distance in terms of an expectation in the following way:

Definition 1.2 (Wasserstein Distance) *For $U,V \in \mathsf{P}(\mathsf{W})$, the p-Wasserstein Distance is de-*

fined as:

$$W_p(U,V) \triangleq \inf_{\gamma} \mathbb{E}_{(X,Y)\sim\gamma}[d(X,Y)^p]$$
$$\text{s.t. } X \sim U,\ Y \sim V \tag{1.3}$$

Regarding the Monge and Kantorovich problems, we can now provide one of the most important theorems from the optimal transportation literature:

Theorem 1.1 (Existence of Optimal Map and Transport Plan) *Let $U, V \in \mathsf{P}(\mathsf{W})$ be probability measures. If U and V admit densities with respect to the Lebesgue measure, u and v, respectively, and if the cost function has the structure $c(x,y) = h(x-y)$ where h is a strictly convex function, then there exists a unique optimal transportation map S^* that minimizes* (1.1)*, and from S^* there is an associated optimal transport plan γ^* that minimizes Equation* (1.2) *[13, 27, 76]*

From Definition 1.2, we also have the following corollary to Theorem 1.1:

Corollary 1.1 *If U and V are probability measures with densities with respect to the Lebesgue measure, u and v, respectively, then the p-Wasserstein Distance can also be written in terms of the transport mapping S as:*

$$W_p(U,V) = \inf_{S:\mathsf{W}\to\mathsf{W}} \int_{w\in\mathsf{W}} d(w,S(w))^p u(w)dw \tag{1.4}$$
$$\text{s.t.} \quad S\#U = V$$

One assumption often used throughout the optimal transportation literature is that the

cost function is the squared Euclidean distance, giving rise to the 2-Wasserstein Distance:

$$W_2(U,V) = \inf_{S:\mathsf{W}\to\mathsf{W}} \int_{w\in\mathsf{W}} ||w - S(w)||^2 u(w) dw \tag{1.5}$$
$$\text{s.t.} \quad S\#U = V$$

We will leverage Theorem 1.1 and Corollary 1.1 heavily to show that our novel optimal transport approach presented from Chapter 2 and onward has a feasible solution. Furthermore, we will be using the 2-Wasserstein Distance to our advantage in Chapter 3. In summary, we can give the following succinct statement regarding this distance metrict: we denote the minimizer of (1.5) as an optimal transport map S^* that is optimal with respect to the squared Euclidean distance, and the minimum value it achieves is the 2-Wasserstein Distance.

1.2 Modern Methods and Applications

Many optimal transport researchers have preferred to use the Kantorovich problem as the basis of forming a computational solution to the transport problem as the constraints are much easier to account for relative to the Monge problem. Some examples of work solving the Kantorovich problem for a variety of applications can be found in [31, 64, 59, 57, 66].

Several authors then proposed regularized versions of the problem so as to increase the computational efficiency even further, whether it be through entropy-based regularization [20, 28, 71], or L^2-based [22]. Through the use of regularization, many application areas that were originally out of reach became practically and computationally feasible to accommodate, and for this reason, several recent approaches to the problem still rely on regularization for tractibility [69, 9].

To solve for the Kantorovich transport plan, problems are often formulated as discretized approximations of the expectation of the transport cost. That is, one normally assumes they have access to a finite numbers of samples from U and a finite number of samples from V to perform

the optimization, and solve for a coupling in a discrete sense. Because of this, one of the most important distinctions between the Monge transport map and the Kantorovich transport plan is that transport plans do not inherently provide the ability to perform "out-of-sample" mappings; that is, the discretized transport plan is not defined outside of the initial set of samples from U and samples from V. As such, a secondary optimization problem must be solved to approximate the transport map itself for samples not included in the original sample set used to create the transport plan. One of the most common methods of generating a transport map in this way is to perform a barycentric projection given the transport plan [78, 21], and in the specific case of the squared Euclidean distance measure, the barycentric projection is indeed representative of the optimal transport plan [2]. A more recent approach uses deep networks to approximate the barycentric projection, leading to increased scalability in larger problem sizes [69].

Throughout the years that optimal transport has existed as a booming area of theoretical research, it has also found a home in a wide range of practical application areas. Perhaps one of the oldest application areas is image analysis, where optimal transportation distances have historically been used to represent a distance metric between images for the purpose of related-image retrieval, or image registration [46, 64, 32]; in general it can be difficult to represent a distance metric between images in a meaningful way. However, the p-Wasserstein distance (also often called the Earth Mover's Distance in much of the literature) can provide a very natural way to represent the "discrepancy" between two candidate images. Transport has also been used as a means of transforming or blending images [63, 26].

In slightly more machine learning-oriented use-cases, optimal transport has been very prominently used in the realm of domain adaptation [17, 18]. Broadly speaking, in a domain adaptation problem, the high-level assumption is that one wants to train a machine learning model, but the training data originates from a somewhat different domain than the testing/evaluation/live data. In this case, a mapping that transforms the training domain to the testing domain or vice-versa can be particularly useful for still allowing the training data to be

useful in the training process. This is especially practical in situations where data from one domain is more easily acquired than data from another related domain, yet the model is still expected to perform well in either domain come time to deploy it.

Historically speaking, optimal transportation methodologies using the Kantorovich problem to solve for the transport plan, and subsequently barycentric projection to extrapolate the result to the full transport map have been constrained to relatively small dimension, usually in image processing applications where feature spaces range from $\mathbb{R}^3$ to $\mathbb{R}^{30}$ at most. But with recent advances that take advantage of deep networks to parameterize either the barycentric projection, or the transport map directly, larger problems have become feasible to accommodate, such as the generative MNIST model demonstrated in [69], a 784-dimensional problem.

1.2.1 Transport vs. Optimal Transport and Novel Contributions

As we prepare to discuss the specific formulation of the optimal transportation problem that this work focuses on, it is worth noting that our approach is somewhat different from the classical approach stemming from Kantorovich's relaxation of the Monge problem. While the notion of the optimal transport cost is taken into consideration in certain incarnations of our problem (specifically, that discussed in Section 3.4.1 and beyond), our initial formulation that we will be discussing in Chapter 2 begins with the Kullback-Leibler Divergence as an objective function, a significantly different approach to the transport problem than the techniques cited in this section. However, as we will show throughout this dissertation, this formulation can lead to further problem formulations that do indeed take the optimal transport cost into account, bridging the gap between the KL-Divergence objective, and more traditional optimal transport methods.

We will conclude this introductory chapter by summarizing the novel contributions of this dissertation for which the dissertation author was the primary investigator. Sections 3.3 and 3.4 begin the discussion of novel developments with two large modifications made to our

group's previously published optimal transport framework in [48, 51], as well as theoretical discussions of the implications of both modifications. Chapter 4 takes a more implementation-based approach to the discussion, describing several strategies that were used to make coding of the novel optimal transport problem more convenient. While the original implementations of our transport problem were constrained to running on CPUs, Section 4.4 takes advantage of a new CUDA-based implementation and provides evaluation metrics of solving the transport problem on different GPU architectures, and offers a few insights into the practical hardware scalability of the algorithm. Section 5 describes an array of applications of the various transport formulations. The density estimation problem (Section 5.2, [16]), generative modeling example (Section 5.3, [52]), and partially-completed reinforcement learning application (Section 5.4), all used the new GPU implementation. The brain-computer interface application (Section 5.1, [73]) used a previously coded CPU implementation of the transport problem, as it was chronologically completed prior to the development of the content of Sections 3.3 and onward.

Chapter 2

Kullback-Leibler-based Measure Transport

In this chapter, we review a version of the optimal transportation problem utilizing the Kullback-Leibler divergence as the basis of the objective function originally described in previous work from our group [40, 42, 41]. This work was also inspired by Monge's problem, as well as further previous work in [25]. This formulation will serve as the starting point for the novel modifications discussed in subsequent sections of this dissertation, and as the basis for the parallelized problems in Chapter 3.

2.1 Kullback-Leibler Divergence as an Objective Function

We recall that the goal of optimal transportation is to identify a transport map S that pushes probability measure U to probability measure V, or in other words, a map S such that $S\#U = V$ in accordance with Definition 1.1. We now present the definition of a diffeomorphism as follows:

Definition 2.1 (Diffeomorphism) *A map $S : \mathsf{W} \to \mathsf{W}$ is a diffeomorphism if S is invertible, and both S and S^{-1} are differentiable.*

For any such diffeomorphism that solves the optimal transportation problem, the following Jacobian equation holds for u and v:

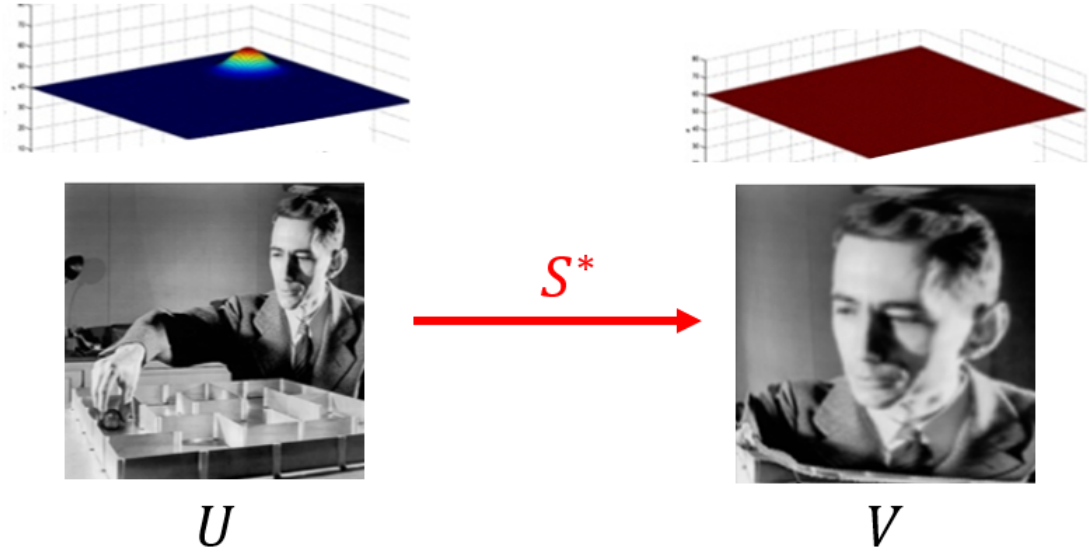


Figure 2.1. A visual representation of the effect of a transport map on the underlying space, when U has a standard Gaussian in its upper-right quadrant, and V is a uniform distribution

$$u(x) = v(S(x))|\det J_S(x)| \tag{2.1}$$

where $J_S(x)$ is the Jacobian of the map S evaluated at the point x. From a conceptual perspective, the Jacobian can be thought of as a representation of how the mapping "warps" space around the point it is evaluating in order to facilitate the desired mapping from U to V. Figure 2.1 shows an example of how the map can potentially warp the domain of a probability distribution. In this example, U has a standard Gaussian distribution centered in the top-right quadrant of the space, and V is a uniform distribution. The image of Shannon represents the shape of the space; as the map pushes U to V, the probability mass in the upper-right quadrant of U gets "stretched" to redistribute the mass to warp the space to a uniform-like shape, resulting in the stretching effect seen in the right-hand image of Shannon.

Furthermore, not only can it be shown that there exists a diffeomorphism that can achieve the solution to the optimal transportation problem, but even more specifically, it can be shown that there exists a *monotonic* diffeomorphism that achieves the solution [76].

Definition 2.2 (Monotonic Diffeomorphism) *A diffeomorphism S is considered monotonic, or orientation-preserving, if $J_S(w) \succeq 0$ for all $w \in \mathsf{W}$.*

Thus, when searching for solutions for the optimal transportation problem, we can safely restrict our search to a feasible set composed of only monotonic diffeomorphisms, and we will denote this set as $\mathfrak{D}_+$. With this smaller feasible set, we can also make a modification to (2.1) by removing the absolute value around the determinant in the specific case of monotonic mappings, and the monotonic Jacobian becomes:

$$u(x) = v(S(x)) \det J_S(x) \tag{2.2}$$

This simplification will indeed become extremely useful as we move forward. If we consider (2.2), we see that the density u is intentionally left by itself on the left-hand side of the equation. We can interpret this in the following way: if we assume V to be a fixed probability measure, then for any candidate mapping $\tilde{S}$ that pushes to V that may or may not solve the optimal transportation problem, we induce a candidate measure $\tilde{U}$ that has a density $\tilde{u}$. This candidate measure $\tilde{U}$ may or may not be the actual U that we wish to push to V, but by interpreting $\tilde{U}$ as induced by $\tilde{S}$, $\tilde{S}$ becomes a free variable over which we can perform optimization. We can then consider the Kullback-Leibler (KL) divergence as a means of measuring the "distance" between $\tilde{U}$ and U.

Definition 2.3 (Kullback-Leibler Divergence) *For $U, V \in \mathsf{P}(\mathsf{W})$, the Kullback-Leibler Divergence, or relative entropy, is a measure of deviation between two probability distributions, and is given as*

$$D(U||V) = \mathbb{E}_U\left[\log \frac{u(X)}{v(X)}\right] \tag{2.3}$$

With this definition in hand, we can then interpret the optimal transport problem in our context as a minimization over all possible $\tilde{S}$ of the KL-Divergence between U and $\tilde{U}$:

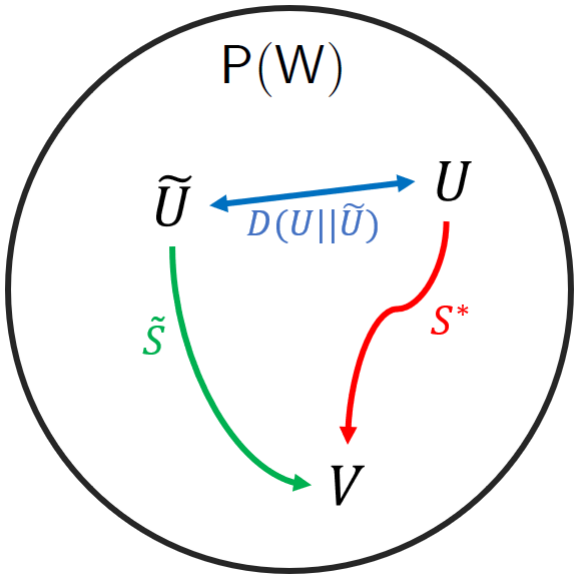


Figure 2.2. A graphical representation of the Kullback-Leibler-based objective function. Our goal is to minimize the KL-Divergence between U and $\tilde{U}$ over $\tilde{S}$. When $D(U||\tilde{U})$ is zero, we have found the correct map, S^*.

$$\min_{\tilde{S}\in\mathfrak{D}_+} D(U||\tilde{U}(\cdot;\tilde{S})) \tag{2.4}$$

where the notation $\tilde{U}(\cdot;\tilde{S})$ is used to make clear the fact that $\tilde{U}$ can be seen as an induced function of a candidate mapping $\tilde{S}$. Because of the fact that the KL-Divergence is exactly zero only when the two arguments represent exactly the same probability distribution, this minimization will ensure that we will be targeting a $\tilde{U}$ that is exactly U. We will once again denote the solution to (2.4) that successfully pushes U to V as S^*. Figure 2.2 depicts an abstract graphical representation of the logic behind this objective function. Here, the circle represents $\mathsf{P}(\mathsf{W})$, of which U, $\tilde{U}$, and V are members. Interpreting V as a pre-set probability distribution that doesn't change, our goal is to optimize over $\tilde{S}$ until $\tilde{U}$ is precisely U. The blue line represents the KL-Divergence between $\tilde{U}$ and U, and when the KL becomes zero, we conclude that $\tilde{U} = U$, and $\tilde{S} = S^*$

From here, we are only a few derivations away from the formulation we will be focusing

on for the duration of the dissertation. If we assume that the expected value of the log of the push-from density u is finite, we can combine (2.4) and (2.2) in the following way:

$$\min_{\tilde{S}\in\mathfrak{D}_+} D(U||\tilde{U}(\cdot;\tilde{S})) \tag{2.5}$$

$$= \min_{\tilde{S}\in\mathfrak{D}_+} \mathbb{E}_U\Big[\log\frac{u(X)}{\tilde{u}(X;\tilde{S})}\Big] \tag{2.6}$$

$$= \min_{\tilde{S}\in\mathfrak{D}_+} -h(u) - \mathbb{E}_U[\log\tilde{u}(X;\tilde{S})] \tag{2.7}$$

$$= \min_{\tilde{S}\in\mathfrak{D}_+} -\mathbb{E}_U[\log\tilde{u}(X;\tilde{S})] \tag{2.8}$$

$$= \min_{\tilde{S}\in\mathfrak{D}_+} -\mathbb{E}_U[\log v(\tilde{S}(X)) + \log\det J_{\tilde{S}}(X)] \tag{2.9}$$

Here, $h(u)$ represents the Shannon differential entropy of u, and we get (2.8) from (2.7) as h is fixed with respect to $\tilde{S}$ and KL-Divergence is non-negative; we then get (2.9) from (2.8) by combining with (2.2). To take the analysis one step further still, if we also make the assumption that *v is log-concave*, we can state the following theorem about Equation (2.9), previously proven in [41, 51]:

Theorem 2.1 (Convex Push-Forward) *If $U,V \in \mathsf{P}(\mathsf{W})$ admit densities u and v, respectively, and if $\mathbb{E}[|\log v(X)|] < \infty$, and if v is log-concave, then:*

$$\min_{\tilde{S}\in\mathfrak{D}_+} -\mathbb{E}_U[\log v(\tilde{S}(X)) + \log\det J_{\tilde{S}}(X)]$$

is a convex optimization problem.

To ease computation of (2.9), our general approach will be to make a Monte Carlo approximation of the expectation with samples drawn from U. Putting all the pieces together, we arrive at a sample-based version of the problem that we can solve, which we succinctly

express here as an argmin over the feasible set of positive semidefinite diffeomorphisms given samples $X_i \sim U$ for $i = 1 \ldots N$:

$$S^* = \arg\min_{\tilde{S} \in \mathfrak{D}_+} \frac{1}{N} \sum_{i=1}^{N} [-\log v(\tilde{S}(X_i)) - \log\det(J_{\tilde{S}}(X_i))] \tag{2.10}$$

In general, to solve the problem we only need *three things*:

1. N independent samples from U, where N is sufficiently large to ensure a good approximation of the expectation
2. Log-concavity of v
3. Knowledge of v up to a normalization constant

Note that without item 2 above, the problem can still potentially be solved, although the objective function becomes nonconvex and one runs the risk of arriving at poor solutions associated with suboptimal local minima.

Nevertheless, given these three components, we can effectively guarantee that Equation (2.9) is convex, and effectively solve for our optimal map. It is also worth noting that knowledge of distribution U is actually not necessary, as long as we have independent samples drawn from it. This is particularly effective if U represents a probability distribution from which we observe samples from the real world, say, in an experiment or other data-gathering scenario. Another notable degree of freedom we have with respect to U is that U need not have any particular structure (e.g. log-concavity) to ensure that (2.9) is a convex optimization problem. We will indeed be using these facts to our advantage for several of the applications discussed in Chapter 5.

One key property to note about the KL-Divergence objective is that it does not actually aim to find the *optimal* transport map, as there is no notion of distance between a source point from U and a destination point from V within the objective (e.g. the Euclidean distance from

the original Monge problem). More accurately stated, the objective merely encodes a problem to seek *a* transport map that successfully pushes U to V. That being said, Section 3.4.1 will augment this objective function to take into account the actual optimal transport cost with respect to the Euclidean distance, and discuss the implications this has for sequential mapping as well.

2.2 Parameterization of the Transport Map

In order to make computation tractable, we must designate a means by which we wish to parameterize the transport map $\tilde{S}$. For this purpose, we will use a linear combination of multivariate polynomial basis functions, or polynomial chaos expansion, as in several instances of previous work [40, 25, 51, 52, 16]. That is, given some vector $x = (x_1, x_2, \ldots, x_d, \ldots, x_D) \in \mathsf{W}$, we can form a multivariate basis function $\phi_{\mathbf{j}}(x)$, where $\mathbf{j}$ represents a multi-index of length D, $\mathbf{j} = (j_1, j_2, \ldots, j_d, \ldots, j_D)$, using a product of D-many univariate polynomials ψ_{j_d} of degree j_d:

$$\phi_{\mathbf{j}}(x) = \prod_{d=1}^{D} \psi_{j_d}(x_d) \tag{2.11}$$

This allows us to represent one component of an arbitrary map $S \in \mathfrak{D}_+$ as a weighted linear combination of the basis functions with coefficients $b_{d,\mathbf{j}}$ as:

$$S^d(x) = \sum_{\mathbf{j} \in \mathscr{J}} b_{d,\mathbf{j}} \phi_{\mathbf{j}}(x) \tag{2.12}$$

where $\mathscr{J}$ represents the set of multi-indices in the representation specifying the order of the polynomials in the associated expansion, and d denotes the d^{th} component of the mapping. In order to make this problem finite-dimensional, we must truncate the expansion to some fixed maximum-order O:

$$\mathscr{J} = \left\{ \mathbf{j} \in \mathbb{N}^D : \sum_{i=1}^{D} j_i \leq O \right\} \tag{2.13}$$

We can now approximate any nonlinear function $S \in \mathfrak{D}_+$ as:

$$S(x) = B\Phi(x) \tag{2.14}$$

where $K \triangleq |\mathscr{J}|$ is the size of the index set and the number of basis functions being used in the expansion, $\Phi(x) = [\phi_{\mathbf{j}_1}(x), \ldots, \phi_{\mathbf{j}_K}(x)]^T]$ is a K-length vector of basis function evaluations at the evaluation point x, and $B \in \mathbb{R}^{D \times K}$ is a matrix of coefficients.

We can now give a finite-dimensional version of (2.10) as:

$$B^* = \arg\min_{\tilde{B}} \frac{1}{N} \sum_{i=1}^{N} [-\log \nu(\tilde{B}\Phi(X_i)) - \log\det(\tilde{B}J_\Phi(X_i))] \tag{2.15}$$

$$\text{s.t.} \quad \tilde{B}J_\phi(X_i) \succeq 0, \; i = 1, \ldots, N \tag{2.16}$$

where $J_\phi(x) = \left[\frac{\partial \phi_{\mathbf{j}_i}}{\partial x_j}(x)\right]_{i,j}$ is the Jacobian matrix of the basis evaluated at the point x. Note that in (2.15), we have modified our optimization problem to find the argmin over the space of coefficient matrices that parameterize our transport map.

In terms of selecting which polynomial basis to use, there are several ways to make a decision. In applications where U is assumed known, the basis functions are usually chosen to be orthogonal with respect to U [81, 67]:

$$\int_{\mathsf{W}} \phi_{\mathbf{j}}(x)\phi_{\mathbf{i}}(x)u(x)dx = \mathbf{1}_{\mathbf{i}=\mathbf{j}} \tag{2.17}$$

where $\mathbf{1}$ is the indicator function. However, in practice, one can technically use any basis of polynomials, and as long as the truncation length K is sufficiently large, the approximation of the parameterized map would theoretically suffice. And with this parameterization of the transport map, we have a complete, finite-dimensional version of the optimization problem that can be actually be solved in a computationally efficient manner.

2.3 Acknowledgments

This chapter featured the reprint of much of the text in “A distributed framework for the construction of transport maps”, Mesa, Diego A.; Tantiongloc, Justin; Mendoza, Marcela; Coleman, Todd P., which has been submitted to Neural Computation in 2018, [52]. The dissertation author was a primary co-investigator of this work.

Chapter 3

Scalability and Regularization of the Optimal Transport Problem

In this chapter, we will investigate two modifications to the Kullback-Leibler Divergence based optimal transportation framework presented in Section 2.1: addressing the issue of scalability by reformulating the problem as a distributed computation via the Alternating Directions Method of Multipliers, and adding a regularization term to the objective via the Wasserstein metric. In addition, we will present a theoretical and computational discussion on how the Wasserstein regularization term plays a role in using compositions of several maps in sequence [51, 52].

3.1 A Global Consensus Problem

We begin our analysis by once again looking at the sample-based optimal transportation problem we derived in Section 2.1, Equation (2.15):

$$B^* = \arg\min_{\tilde{B}} \frac{1}{N}\sum_{i=1}^{N}[-\log \nu(\tilde{B}\Phi(X_i)) - \log\det(\tilde{B}J_\Phi(X_i))]$$
$$\text{s.t.} \quad \tilde{B}J_\phi(X_i) \succeq 0,\ i = 1,\ldots,N$$

Upon examination of (2.15), we see that this optimization problem is, more generally

speaking, of the form

$$\min_x \sum_{i=1}^{N} f_i(x) \tag{3.1}$$

To be more clear, the optimization problem we would like to solve is a sum of N-many objective terms, where each of the objective terms is convex (under the assumption that v is log-concave), and each objective term is a function of a single variable over which we would like to perform optimization, $\tilde{B}$. The key insight here is that problems of this form are often separated into N-many independent optimization problems that can be carried out in parallel by using a *global consensus* strategy instead; in other words, rather than treat the optimization variable as a single variable shared across each term of the objective, we instead optimize over N-many optimization variables, but impose a constraint that they must ultimately be identical when a solution is found [8, 11]:

$$\min_{x_1,x_2,\ldots,x_N,z} \sum_{i=1}^{N} f_i(x_i) \tag{3.2}$$

$$\text{s.t. } x_i - z = 0, \quad i = 1,\ldots,N \tag{3.3}$$

Here, the optimization is now separable across the summation, as we've introduced a separate optimization variable for each of the N objective functions, and added the consensus constraint across them. With this in mind, we can now write a global consensus version of (2.15):

$$\min_{\tilde{B}_i, B} \frac{1}{N} \sum_{i=1}^{N} [-\log v(\tilde{B}_i \Phi(X_i)) - \log \det(\tilde{B}_i J_\Phi(X_i))]$$

$$\text{s.t.} \quad \tilde{B}_i J_\phi(X_i) \succeq 0, \; i = 1, \ldots, N$$

$$\tilde{B}_i = B, \quad i = 1, \ldots, N \tag{3.4}$$

Here we have used somewhat of a shorthand notation for the optimization variables, as the optimization is over $\tilde{B}_i$ for all $i = 1, \ldots, N$, as well as B; in other words, we have $N+1$ many optimization variables in this problem, and the final solution, B^*, will be the final value of B. As another interpretation, we can also think of each sample, X_i, as independently inducing some random $\tilde{U}_i$ through each of their associated maps $\tilde{B}_i$, and the act of solving this problem can be thought of as iteratively reducing the distance between each $\tilde{U}_i$ and the true U by reducing the distance between each $\tilde{B}_i$.

3.2 Parallelization via ADMM

With the global consensus form of the problem, we can now reformulate (3.4) within the framework of the Alternating Direction Method of Multipliers (ADMM) [51]. In general, ADMM is an iterative strategy that allows for the decomposition of large optimization problems involving many optimization variables into separate local sub-problems which are much easier to solve individually, and can further be solved sequentially with their results combining to yield the solution to the overall joint optimization problem. One of the most comprehensive reviews and treatments of the strategy can be found in this work by Boyd [11], but the general steps are outlined as follows.

Under mild assumptions, ADMM not only allows for sequential decomposition of an optimization problem over separate variables, but it is also guaranteed to converge with sufficient iterations. The most general form of a problem amenable to ADMM is the following:

$$\min_{x,z} f(x) + g(z) \tag{3.5}$$

$$\text{s.t.} \quad Ax + Bz = c \tag{3.6}$$

where $Ax + Bz = c$ is a set of constraints. We then form the Lagrangian by raising the constraints to the objective and adding an associated vector of dual variables, λ:

$$\min_{x,z} f(x) + g(z) + \lambda^T (Ax + Bz - c) \tag{3.7}$$

From the Lagrangian, we then form the penalized form of the optimization as:

$$L_\rho(x, z, \lambda) = f(x) + g(z) + \lambda^T (Ax + Bz - c) + \frac{\rho}{2} ||Ax + Bz - c||_2^2 \tag{3.8}$$

where $\rho > 0$ is a penalty parameter. In this form, ADMM then dictates that the optimization can be performed by alternating between solving for the variables x and z in sequence, followed by an update to the dual variables, the latter of which is dependent on the penalty parameter ρ which can be interpreted as a step size for the dual update:

$$x^{k+1} = \arg\min_x L_\rho(x, z^k, \lambda^k) \tag{3.9}$$

$$z^{k+1} = \arg\min_z L_\rho(x^{k+1}, z, \lambda^k) \tag{3.10}$$

$$\lambda^{k+1} = \lambda^k + \rho(Ax^{k+1} + Bz^{k+1} - c) \tag{3.11}$$

where the k superscript denotes iteration index. Under relatively mild assumptions,

ADMM guarantees that in this form, as $k \to \infty$, the values of x^k and z^k will converge to the true solution to the original joint optimization problem [11].

Using this framework, we can now derive the parallelized version of (3.4), following previous work from our research group [51, 52]. Moving forward, for notational clarity, we will write Φ_i instead of $\Phi(X_i)$, and J_i instead of $J_\Phi(X_i)$. We also introduce the following auxiliary variables:

- $B\Phi_i \triangleq p_i$: represents the optimal transport transformation of a point X_i using the current consensus variable
- $BJ_i \triangleq Z_i$: represents the evaluation of the consensus-variable-based Jacobian at the point X_i

We can now write the constrained version of 3.4 as:

$$\min_{\tilde{B}_i, Z_i, p_i, B} \frac{1}{N} \sum_{i=1}^{N} -\log v(p_i) - \log\det Z_i \tag{3.12}$$

$$\begin{aligned} \text{s.t.} \quad & B\Phi_i = p_i : \qquad \gamma_i \quad (D \times 1) \\ & BJ_i = Z_i : \qquad \lambda_i \quad (D \times D) \\ & \tilde{B}_i - B = 0 : \qquad \alpha_i \quad (D \times K) \\ & Z_i \succ 0 \end{aligned}$$

where we have denoted the dual variable that will be associated with each constraint, as well as its dimensionality to the right of the constraint itself. And now the final step that remains is to form the fully-penalized Lagrangian as:

$$L(\tilde{B}_i, Z_i, p_i, B, \gamma_i, \lambda_i, \alpha_i) = \frac{1}{N}\sum_{i=1}^{N} -\log v(p_i) - \log\det Z_i \tag{3.13}$$
$$+ \frac{1}{N}\sum_{i=1}^{N} \gamma_i^T (p_i - B\Phi_i) + \mathbf{tr}(\lambda_i^T (Z_i - BJ_i)) + \mathbf{tr}(\alpha_i^T (\tilde{B}_i - B))$$
$$+ \frac{1}{N}\sum_{i=1}^{N} \frac{\rho}{2}||\tilde{B}_i - B||_2^2 + \frac{\rho}{2}||B\Phi_i - p_i||_2^2 + \frac{\rho}{2}||BJ_i - Z_i||_2^2$$

To make one aspect of this notation slightly more clear, note that the penalized Lagrangian is actually a function of N-many of each optimization variable with the exception of the consensus variable, hence the subscript i for these variables. However, each i^{th} instance of each variable is only dependent on the i^{th} instance of the other variables, which will allow us to ultimately parallelize the computation across N-many threads, a more detailed discussion of which will be presented in the following section.

ADMM then allows us to solve for each of these variables in sequence using only the most recent versions of each in subsequent optimization sub-problems. Echoing the ADMM update format from earlier, we can represent the sequential updates for our problem in the

following way:

$$B^{k+1} = \arg\min_{B} L(\tilde{B}_i^k, Z_i^k, p_i^k, B, \gamma_i^k, \lambda_i^k, \alpha_i^k) \tag{3.14}$$

$$\tilde{B}_i^{k+1} = \arg\min_{\tilde{B}_i} L(\tilde{B}_i, Z_i^k, p_i^k, B^{k+1}, \gamma_i^k, \lambda_i^k, \alpha_i^k) \tag{3.15}$$

$$Z_i^{k+1} = \arg\min_{Z_i} L(\tilde{B}_i^{k+1}, Z_i, p_i^k, B^{k+1}, \gamma_i^k, \lambda_i^k, \alpha_i^k) \tag{3.16}$$

$$p_i^{k+1} = \arg\min_{p_i} L(\tilde{B}_i^{k+1}, Z_i^{k+1}, p_i, B^{k+1}, \gamma_i^k, \lambda_i^k, \alpha_i^k) \tag{3.17}$$

$$\gamma_i^{k+1} = \gamma_i^k + \rho(p_i^{k+1} - B^{k+1}\Phi_i) \tag{3.18}$$

$$\lambda_i^{k+1} = \lambda_i^k + \rho(Z_i^{k+1} - B^{k+1}J_i) \tag{3.19}$$

$$\alpha_i^{k+1} = \alpha_i^k + \rho(\tilde{B}_i^{k+1} - B^{k+1}) \tag{3.20}$$

3.2.1 Closed Form Updates

Upon closer examination of the penalized Lagrangian, we can actually derive closed-form solutions for the updates for B^{k+1}, $\tilde{B}_i^{k+1}$, and Z_i^{k+1}. In this section, we will describe these derivations in more detail.

The consensus variable, B^{k+1}, has an associated loss function stemming from the penalized Lagrangian of:

$$\begin{aligned} C(B) = \frac{1}{N}\sum_{i=1}^{N} \gamma_i^T (p_i - B\Phi_i) + \mathbf{tr}(\lambda_i^T (Z_i - BJ_i)) + \mathbf{tr}(\alpha_i^T (\tilde{B}_i - B)) \\ + \frac{1}{N}\sum_{i=1}^{N} \frac{\rho}{2}||\tilde{B}_i - B||_2^2 + \frac{\rho}{2}||B\Phi_i - p_i||_2^2 + \frac{\rho}{2}||BJ_i - Z_i||_2^2 \end{aligned} \tag{3.21}$$

An important thing to note is that this loss function is necessarily convex, and as such we can derive a closed form solution fairly easily. We first take the first-order derivative of this expression to arrive at:

$$\frac{\partial C(B)}{\partial B} = \frac{1}{N}\sum_{i=1}^{N} -\gamma_i^k \Phi_i^T - \lambda_i^k J_i - \alpha_i^{kT} + \frac{1}{N}\sum_{i=1}^{N} \rho(BJ_i - Z_i^k)J_i^T + \rho(B - \tilde{B}_i^k) + \rho(B\Phi_i - p_i^k)\Phi_i^T \tag{3.22}$$

We then set this equation to zero, and express it in terms of B:

$$B\Big[\rho\Big(I + \frac{1}{N}\sum_{i=1}^{N}\Phi_i\Phi_i^T + J_iJ_i^T\Big)\Big] = \frac{1}{N}\sum_{i=1}^{N}[\rho(\tilde{B}_i^k + p_i^k\Phi_i^T + Z_i^kJ_i^T) + \gamma_i^k\Phi_i^T + \lambda_i^kJ_i^T + \alpha_i^k] \tag{3.23}$$

If we then define

$$L \triangleq \Big[\rho\Big(I + \frac{1}{N}\sum_{i=1}^{N}\Phi_i\Phi_i^T + J_iJ_i^T\Big)\Big]$$

and

$$M \triangleq \frac{1}{N}\sum_{i=1}^{N}[\rho(\tilde{B}_i^k + p_i^k\Phi_i^T + Z_i^kJ_i^T) + \gamma_i^k\Phi_i^T + \lambda_i^kJ_i^T + \alpha_i^k]$$

then the closed form solution for B^{k+1} is simply $M \cdot L^{-1}$.

For $\tilde{B}_i$, we can form a similar loss function:

$$C(\tilde{B}_i) = \frac{\rho}{2}||\tilde{B}_i - B^{k+1}|| + \mathbf{tr}(\alpha_i^{kT}(\tilde{B}_i - B^{k+1})) \tag{3.24}$$

Taking the first-order derivative of this loss function with respect to $\tilde{B}_i$, setting to zero, and solving for $\tilde{B}_i$, we arrive at a closed-form update of:

$$\tilde{B}_i^{k+1} = -\frac{1}{\rho}\alpha_i^k + B^{k+1} \tag{3.25}$$

Finally, we can derive the slightly more involved closed-form update for Z_i as follows. We refer to [12] to derive the first-order optimality condition for the loss function associated with Z_i as

$$\begin{aligned} &-Z_i^{-1} + \rho(Z_i - B^{k+1}J_i) + \lambda_i^k = 0 \\ &\rightarrow \rho Z_i - Z_i^{-1} = \rho B^{k+1}J_i - \lambda_i^k \end{aligned} \tag{3.26}$$

We then take the eigenvalue decomposition of the right hand side of (3.26):

$$\rho B^{k+1}J_i - \lambda_i^k = Q\Lambda Q^T \tag{3.27}$$

where $\Lambda = \mathbf{diag}(v_1, \ldots, v_D)$, and $Q^T Q = QQ^T = I$. Multiplying (3.26) by Q^T on the left and by Q on the right gives us the following:

$$\rho\tilde{Z}_i - \tilde{Z}_i^{-1} = \Lambda \tag{3.28}$$

where $\tilde{Z} = Q^T Z_i Q$. Furthermore, a diagonal solution to this equation is given by:

$$\tilde{Z}_{i,(jj)} = \frac{v_j + \sqrt{v_j^2 + 4\rho}}{2\rho} \quad j = 1, \ldots, D \tag{3.29}$$

Note that to enforce the positive semidefiniteness of Z_i, we specifically take the positive root of this equation. Then finally, our closed-form Z_i update becomes

$$Z_i^{k+1} = Q\tilde{Z}_i Q^T \tag{3.30}$$

3.2.2 The p_i-update

The final ADMM variable update we need to consider is the update for p_i, Equation (3.17). If we look at the loss function associated with p_i, we get

$$C(p_i) = -\log v(p_i) + \frac{\rho}{2}||B^{k+1}\Phi_i - p_i||_2^2 + \gamma_i^{kT}(p_i - B^{k+1}\Phi_i) \tag{3.31}$$

Unlike the other optimization variables, the existence of a closed-form solution for the p_i update is dependent on the specific structure of the distribution v. Therefore, the p_i update can potentially necessitate other optimization procedures to solve, such as a gradient-based method, under certain circumstances. In Chapter 5, we will demonstrate both an instance of when the p_i update can in fact be solved in closed form (when v is a Gaussian density), and when it cannot (Bayesian inference with a logistic likelihood).

3.2.3 Optimizing the Structure of the Transport Map

An important consideration in ensuring that the computation of a transport map using the above methodology is efficient is the underlying structure of the map parameterization. In Section 2.2, we presented a parameterization of the transport map utilizing a polynomial chaos expansion and the multi-index set $\mathscr{J}$, each member of which denoted the orders of univariate polynomial terms involved in the product for each multivariate basis term. For this fully-expressive mapping, each component of the output of the mapping, denoted S^d, is a function of every component of the input point, like so:

$$S(x_i) = \begin{bmatrix} S^1(x_i^1, x_i^2, \ldots, x_i^D) \\ S^2(x_i^1, x_i^2, \ldots, x_i^D) \\ \vdots \\ S^D(x_i^1, x_i^2, \ldots, x_i^D) \end{bmatrix} \tag{3.32}$$

However, this parameterization tends to be infeasible to use in high dimension (D is very high) or with exceedingly high order polynomials (O is very high) due to the exponential rate at which the number of polynomials increases with respect to D and O. In terms of the basis length $K = |\mathscr{J}|$, the number of terms for the fully-expressive mapping is $\binom{D+O}{D}$. Combined with the number of optimization variables required to solve the ADMM formulation, the fully-expressive mapping caused out-of-memory errors for dimensions as low as $D = 10$ to 15, depending on the size of N, drastically limiting our ability to apply the framework to realistically-sized problems.

To the end of easing computation in this manner, we refer to [49], where two less expressive, but more computationally feasible, map structures were discussed. In this section, we will reproduce the discussion of these two other parameterization, manifesting in two different multi-index sets that can be used alternatively to $\mathscr{J}$.

The first alternative to the fully-expressive map is the Knothe-Rosenblatt (KR) map [10], which our group also previously used within the context of generating transport maps for optimal message point feedback communication [48]. For the KR map, each d^{th} component of the output is only a function of the first d components of the input, resulting in a *lower-triangular mapping*, and thus the mapping has the following structure:

$$S(x_i) = \begin{bmatrix} S^1(x_i^1) \\ S^2(x_i^1, x_i^2) \\ \vdots \\ S^D(x_i^1, x_i^2, \ldots, x_i^D) \end{bmatrix} \tag{3.33}$$

Both the KR and fully-expressive mapping described before perform the transport from one density to another, but with different geometric transformations. An example of these differences can be found in Figures 3 and 4 of [48].

A KR arrangement gives rise to the following modified multi-index set (note that the multi-index set is now sub-scripted according to the dimension of the data, as membership in

the set is dependent on the data component):

$$\mathscr{J}_d^{KR} = \left\{ \mathbf{j} \in \mathbb{N}^D : \sum_{i=1}^{D} j_i \leq O \wedge j_i = 0, \forall i < d \right\}, d = 1, \ldots, D \tag{3.34}$$

However, even the full KR mapping results in $\binom{D+O}{D} - D$ basis terms, and so the size complexity of the parameterization can indeed still be a problem.

An even less expressive parameterization still is a KR mapping that ignores all multivariate polynomials that are a function of more than one data component of the input, resulting in the following multi-index set:

$$\mathscr{J}_d^{KRSV} = \left\{ \mathbf{j} \in \mathbb{N}^D : \sum_{i=1}^{D} j_i \leq O \wedge j_i j_l = 0, \forall i \neq l \wedge j_i = 0, \forall i > d \right\}, \quad d = 1, \ldots, D \tag{3.35}$$

We will refer to this parameterization as the "Single Variable" Knothe-Rosenblatt mapping, which itself only results $DO + 1$ total basis terms, making it by far the most computationally tractable, albeit the least effective function approximation. That being said, however, we will be presenting several applications in Chapter 5 that do in fact take advantage of the Single Variable KR mapping, and still manage to achieve relatively good performance at the corresponding machine learning task.

3.3 A Knothe-Rosenblatt Optimal Transportation Problem

An especially useful property of the KR parameterization, applicable to both the full KR and Single Variable mappings, is the following identity for the Jacobian of the map:

$$\log\det(J_S(X_i)) = \sum_{d=1}^{D} \log \partial_d S^d(X_i) \tag{3.36}$$

where $\partial_d S^d(X_i)$ represents the partial derivative of the d^{th} component of the mapping

with respect to the d^{th} component of the data, evaluated at the point X_i. Furthermore, the positive-semidefiniteness of the Jacobian can equivalently be enforced for a lower-triangular mapping by ensuring the following:

$$\partial_d S^d > 0, \quad 1 \leq d \leq D \tag{3.37}$$

We are then in a position to write a Knothe-Rosenblatt special-case version of (3.4). To this end, we first introduce a few new variables to assist with the notation (as well as the imminent conversion of the objective function to an ADMM problem):

- Φ_i^d represents the partial derivative of Φ_i taken with respect to the d^{th} component of the input data. Therefore, $\tilde{B}\Phi_i^d = \partial_d S(X_i) \in \mathbb{R}^D$, and $\partial_d S^d(X_i)$ is simply the d^{th} component of $\tilde{B}\Phi_i^d$.
- $\mathbf{1}_d$ represents a one-hot vector of length D with the one in the d^{th} position
- $Y_i^d \triangleq \tilde{B}\Phi_i^d$

Using this new notation, we can fully express the Knothe-Rosenblatt special-case of (3.4) as:

$$\begin{aligned}
&\min_{\tilde{B}_i} \frac{1}{N}\sum_{i=1}^{N}[-\log \nu(\tilde{B}_i\Phi(X_i)) - \sum_{d=1}^{D}\log Y_i^d \mathbf{1}_d] \\
&\text{s.t.} \quad Y_i^d \mathbf{1}_d > 0,\ i = 1,\ldots,N,\ d = 1,\ldots,D \\
&\tilde{B}\Phi_i^d = Y_i^d \\
&\tilde{B}_i = B, \quad i = 1,\ldots,N
\end{aligned} \tag{3.38}$$

One very key advantage here to note relative to the original fully-expressive mapping formulation is that the positive-semidefiniteness constraints has been replaced by positive scalar

constraints instead, which are computationally much easier to handle. Indeed, as we continue with the ADMM derivation, we will see that these scalar constraints end up effectively replacing the Eigenvalue decomposition of (3.27) with a scalar quadratic equation evaluation instead, leading to much quicker computation. Furthermore, as a prelude to upcoming sections, it is also worth mentioning at this point that this constraint modification also makes it much easier to implement the ADMM formulation of this problem in CUDA, as Eigenvalue decompositions in large quantities (for example, N-many of them, as the original formulation required) can still be quite the computational bottleneck on GPUs.

3.3.1 Parallelized Knothe-Rosenblatt Optimal Transport

Now we will derive the parallelized, consensus ADMM version of (3.38). This process will, in general, look very similar to that of Section 3.2. We begin by introducing two auxiliary variables (in addition to Y_i^d and Φ_i^d presented in the previous section) to assist with the formulation:

- $\tilde{B}\Phi_i = p_i$: once again, this represents the mapping of a sample X_i through the mapping parameterized by the optimization variable $\tilde{B}$
- $Z_i^d = Y_i^d \mathbf{1}_d$: Z_i^d represents the d^{th} component of Y_i^d. Therefore, Z_i^d is a scalar value, while Y_i^d is a vector of length D. Note that there is somewhat of a notational inconsistency between Y_i^d and Z_i^d; in the former case, the superscript d denotes the fact that Y_i^d is the partial derivative of the mapping with respect to the d^{th} component of the input data, but in the latter case, the superscript d indicates that Z_i^d is the d^{th} component of Y_i^d. Although this may be notationally confusing, it does make further analysis a bit simpler, as the superscript of d is consistently associated with optimization variables now being potentially indexed by d in addition to by i. Furthermore, this is an echoing of the notation used in our previous work as well [52].

With all of these auxiliary variables in place, we rewrite (3.38) as:

$$\begin{aligned}\min_{\tilde{B}_i, p_i, Y_i^d, Z_i^d, B} \; & \frac{1}{N}\sum_{i=1}^{N} -\log v(p_i) + \frac{\rho}{2}||\tilde{B}_i - B||_2^2 + \frac{\rho}{2}||B\Phi_i - p_i||_2^2 \\ & \sum_{d=1}^{D} -\log Z_i^d + \frac{\rho}{2}(Y_i^d \mathbf{1}_d - Z_i^d)^2 + \frac{\rho}{2}||B\Phi_i^d - Y_i^d||_2^2 \end{aligned} \tag{3.39}$$

$$\begin{aligned} \text{s.t. } & B\Phi_i = p_i : \qquad \gamma_i \; (D \times 1) \\ & \tilde{B}_i - B = 0 : \qquad \alpha_i \; (D \times K) \\ & Y_i^d \mathbf{1}_d = Z_i^d : \qquad \beta_i^d \; (1 \times 1) \\ & B\Phi_i^d = Y_i^d : \qquad \lambda_i^d \; (D \times 1) \\ & Z_i^d > 0 \end{aligned} \tag{3.40}$$

where we have denoted the corresponding dual variables and their dimension to the right of each constraint. Once again, we emphasize that the superscript d in this formulation represents the fact that in addition to having variables associated with each data sample indexed by i, some variables are now unique to an index over dimension as well. For example, there are DN-many Z variables that must be solved for in this formulation. We can now raise the constraints to form the fully-penalized Lagrangian as:

$$\begin{aligned} & L_\rho(\tilde{B}_i, Z_i^d, Y_i^d, p_i, B; \gamma_i, \alpha_i, \beta_i^d, \lambda_i^d) \\ & = \frac{1}{N}\sum_{i=1}^{N} -\log v(p_i) + \gamma_i^T(p_i - B\Phi_i) + \mathbf{tr}(\alpha_i^T(\tilde{B}_i - B)) \\ & + \frac{\rho}{2}||\tilde{B}_i - B||_2^2 + \frac{\rho}{2}||B\Phi_i - p_i||_2^2 \\ & + \sum_{d=1}^{D} -\log Z_i^d + \beta_i^d(Z_i^d - Y_i^d \mathbf{1}_d) + \lambda_i^{dT}(Y_i^d - B\Phi_i^d) \\ & + \frac{\rho}{2}(Y_i^d \mathbf{1}_d - Z_i^d)^2 + \frac{\rho}{2}||B\Phi_i^d - Y_i^d||_2^2 \end{aligned} \tag{3.41}$$

Similar to before, the final ADMM update equations for each variable are closed-form, with the exception with the optimization over p_i. We will echo Section 3.2 and begin by showing the sequential order of update equations we use during the optimization process; note that in order to not confuse the superscript d from ADMM iteration index, we will enclose the iteration index k in parentheses for clarity:

$$B^{(k+1)} = \arg\min_{B} L_\rho(\tilde{B}_i^{(k)}, Z_i^{d(k)}, Y_i^{d(k)}, p_i^{(k)}, B; \gamma_i^{(k)}, \alpha_i^{(k)}, \beta_i^{d(k)}, \lambda_i^{d(k)}) \tag{3.42}$$

$$\tilde{B}_i^{(k+1)} = \arg\min_{\tilde{B}_i} L_\rho(\tilde{B}_i, Z_i^{d(k)}, Y_i^{d(k)}, p_i^{(k)}, B^{(k+1)}; \gamma_i^{(k)}, \alpha_i^{(k)}, \beta_i^{d(k)}, \lambda_i^{d(k)}) \tag{3.43}$$

$$Z_i^{d(k+1)} = \arg\min_{Z_i^d} L_\rho(\tilde{B}_i^{(k+1)}, Z_i^d, Y_i^{d(k)}, p_i^{(k)}, B^{(k+1)}; \gamma_i^{(k)}, \alpha_i^{(k)}, \beta_i^{d(k)}, \lambda_i^{d(k)}) \tag{3.44}$$

$$Y_i^{d(k+1)} = \arg\min_{Y_i^d} L_\rho(\tilde{B}_i^{(k+1)}, Z_i^{d(k+1)}, Y_i^d, p_i^{(k)}, B^{(k+1)}; \gamma_i^{(k)}, \alpha_i^{(k)}, \beta_i^{d(k)}, \lambda_i^{d(k)}) \tag{3.45}$$

$$p_i^{(k+1)} = \arg\min_{p_i} L_\rho(\tilde{B}_i^{(k+1)}, Z_i^{d(k+1)}, Y_i^{d(k+1)}, p_i, B^{(k+1)}; \gamma_i^{(k)}, \alpha_i^{(k)}, \beta_i^{d(k)}, \lambda_i^{d(k)}) \tag{3.46}$$

$$\gamma_i^{(k+1)} = \gamma_i^{(k)} + \rho(p_i^{(k+1)} - B^{(k+1)}\Phi_i) \tag{3.47}$$

$$\alpha_i^{(k+1)} = \alpha_i^{(k)} + \rho(\tilde{B}_i^{(k+1)} - B^{(k+1)}) \tag{3.48}$$

$$\lambda_i^{d(k+1)} = \lambda_i^{d(k)} + \rho(Y_i^{d(k+1)} - B^{(k+1)}\Phi_i^d) \tag{3.49}$$

$$\beta_i^{d(k+1)} = \beta_i^{d(k)} + \rho(Z_i^{d(k+1)} - Y_i^{d(k+1)}\mathbf{1}_d) \tag{3.50}$$

3.3.2 Knothe-Rosenblatt Closed-Form Updates and Results

As previously mentioned, we can now derive the closed-form solutions to all of the optimization variable updates, with the exception of p_i. In fact, the general form of the update for p_i remains the same as

$$\arg\min_{p_i} \; -\log v(p_i) + \frac{\rho}{2}||B^{(k+1)}\Phi_i - p_i||_2^2 + \gamma_i^{(k)T}(p_i - B^{(k+1)}\Phi_i)$$

In addition, the closed-form update for $\tilde{B}_i$ remains the same as well:

$$\tilde{B}_i^{(k+1)} = -\frac{1}{\rho}\alpha_i^{(k)} + B^{(k+1)} \tag{3.51}$$

As such, all that remains is to derive the closed-form updates for B, Y_i^d, and Z_i^d.

The cost function associated with B is given by

$$\begin{aligned} C(B) = &\frac{1}{N}\sum_{i=1}^{N}\frac{\rho}{2}||\tilde{B}_i^{(k)} - B||_2^2 + \frac{\rho}{2}||B\Phi_i - p_i^{(k)}||_2^2 \\ &+ \gamma_i^{(k)T}(p_i^{(k)} - B\Phi_i) + \mathbf{tr}(\alpha_i^{(k)T}(\tilde{B}_i^{(k)} - B)) \\ &+ \sum_{d=1}^{D}\frac{\rho}{2}||B\Phi_i^d - Y_i^{d(k)}||_2^2 + \lambda_i^{d(k)T}(Y_i^{d(k)} - B\Phi_i^d) \end{aligned} \tag{3.52}$$

Taking the first order derivative, setting to zero, and putting the equation in terms of B gives us:

$$\begin{aligned} &B\Big[\rho(I + \frac{1}{N}\sum_{i=1}^{N}\Phi_i\Phi_i^T + \sum_{d=1}^{D}\Phi_i^d\Phi_i^{dT})\Big] \\ &= \frac{1}{N}\sum_{i=1}^{N}\rho\tilde{B}_i^{(k)} + \rho p_i^{(k)}\Phi_i^T + \gamma_i^{(k)}\Phi_i^T + \alpha_i^{(k)T} \\ &+ \sum_{d=1}^{D}\rho Y_i^{d(k)}\Phi_i^{dT} + \lambda_i^{d(k)}\Phi_i^{dT} \end{aligned} \tag{3.53}$$

And by letting

$$L = \rho(I + \frac{1}{N}\sum_{i=1}^{N}\Phi_i\Phi_i^T + \sum_{d=1}^{D}\Phi_i^d\Phi_i^{dT}) \tag{3.54}$$

and

$$M = \frac{1}{N}\sum_{i=1}^{N}\rho\tilde{B}_i^{(k)} + \rho p_i^{(k)}\Phi_i^T + \gamma_i^{(k)}\Phi_i^T + \alpha_i^{(k)T} + \sum_{d=1}^{D}\rho Y_i^{d(k)}\Phi_i^{dT} + \lambda_i^{d(k)}\Phi_i^{dT} \tag{3.55}$$

we can once again express the final closed-form update of B as:

$$B^{(k+1)} = M \cdot L^{-1} \tag{3.56}$$

The loss function associated with Z_i^d for a given i and given d is:

$$C(Z_i^d) = -\log Z_i^d + \frac{\rho}{2}(Y_i^{d(k)}\mathbf{1}_d - Z_i^d)^2 \tag{3.57}$$

$$+ \beta_i^{d(k)}(Z_i^d - Y_i^{d(k)}\mathbf{1}_d) \tag{3.58}$$

Relative to the original ADMM formulation, Z_i^d is a scalar value for any given i and any given d. Thus, taking the derivative and setting to zero, we get the following scalar quadratic expression:

$$\rho Z_i^{d2} + (\beta_i^{d(k)} - \rho Y_i^{d(k)}\mathbf{1}_d)Z_i^d - 1 = 0 \tag{3.59}$$

As we would like to enforce the constraint that $Z_i^{d(k+1)}$ be greater than 0, we set the closed-form solution to the positive root of (3.59):

$$Z_i^{d(k+1)} = \frac{\rho Y_i^{d(k)}\mathbf{1}_d - \beta_i^{d(k)} + \sqrt{(\beta_i^{d(k)} - \rho Y_i^{d(k)}\mathbf{1}_d)^2 + 4\rho}}{2\rho} \tag{3.60}$$

Finally, the loss function associated with Y_i^d is:

$$C(Y_i^d) = \frac{\rho}{2}(Y_i^d\mathbf{1}_d - Z_i^{d(k+1)})^2 + \frac{\rho}{2}||B^{(k+1)}\Phi_i^d - Y_i^d||_2^2$$

$$+ \beta_i^{d(k)}(Z_i^{d(k+1)} - Y_i^d\mathbf{1}_d) + \lambda_i^{d(k)T}(Y_i^d - B^{(k+1)}\Phi_i^d) \tag{3.61}$$

Taking the derivative, setting to 0, and putting in terms of Y_i^d, we arrive at the following

closed-form update for Y_i^d:

$$Y_i^{d(k+1)} = (\rho Z_i^{d(k+1)} \mathbf{1}_d^T + \rho B^{(k+1)} \Phi_i^d + \beta_i^{d(k)} \mathbf{1}_d^T - \lambda_i^{d(k)T}) \cdot (\rho \mathbf{1}_d \mathbf{1}_d^T + \rho I)^{-1} \tag{3.62}$$

Using this formulation, as well as these closed-form updates, Figure 3.1 shows the resulting behavior of a mapping designed to push a synthetic 2-dimensional bimodal distribution to a standard Gaussian distribution using the full Knothe-Rosenblatt parameterization. For a map pushing to a Gaussian with mean μ and covariance Σ, the p_i update does admit a closed-form solution, making it a very efficient solve:

$$p_i^{k+1} = (\rho B^{k+1} \Phi_i - \gamma_i^{k+1} + \Sigma^{-1} \mu) \cdot (\Sigma^{-1} + \rho \mathbf{I})^{-1} \tag{3.63}$$

As U was a synthetic distribution in this case, we readily drew "training" samples from U to construct the mapping using our computational framework, which was constructed using a Full Knothe-Rosenblatt parameterization of order 8. We then drew a new batch of "testing" samples from U to evaluate the quality of the computed map. The figure depicts the transport of these testing samples from the perspective of a Kernel Density Estimate of U and V using the non-transported/transported samples, respectively (top row), and from the perspective of the non-transported/transported samples themselves (bottom row). The key thing to notice that while we do not use a fully-expressive mapping, the performance of the map is still quite impressive in terms of transforming samples to the desired V distribution.

For comparison, Figure 3.2 is a similar depiction of the behavior of a Single Variable Knothe-Rosenblatt, order 8 mapping designed to perform the same push from U to V, using the same samples from U to train/test the mapping as was used for the first figure. When comparing the two figures, the loss in transport quality indeed becomes obvious as we move from the full Knothe-Rosenblatt mapping to the Single Variable version. This tradeoff between model complexity and map performance is a very important consideration for any application of this

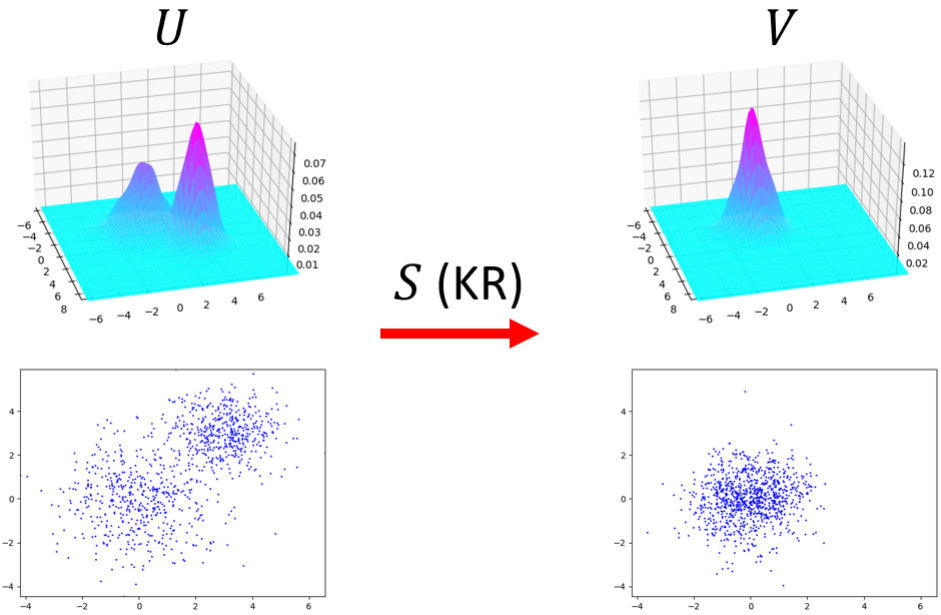


Figure 3.1. An example mapping computed to push a 2-dimensional bimodal distribution to a standard Gaussian distribution. The mapping is depicted from both the perspective of a Kernel Density Estimate of U and V (top), as well as from the perspective of the samples themselves (bottom).

framework, and is highly dependent on use-case constraints, and availability of computational hardware. However, while the Single Variable Knothe-Rosenblatt mapping has a significantly less expressive basis than the Full Knothe-Rosenblatt mapping, it still has very much acceptable performance in many cases, and should not necessarily be ignored as an option for any practical application.

And finally, Figure 3.3 shows a "stress-test" we performed using our full Knothe-Rosenblatt mappings. The purpose of this example is to investigate how the Knothe-Rosenblatt mapping performs in situations where U has highly irregular structure, and even borderline discontinuous density. First, we randomly selected images from the MNIST dataset (one per digit), and treated the greyscale intensities of each image as representative of a probability distribution. Using the greyscale pixel intensities for each image, we then drew samples across each image in 2-dimensional space, and added a small amount of Gaussian noise to each draw

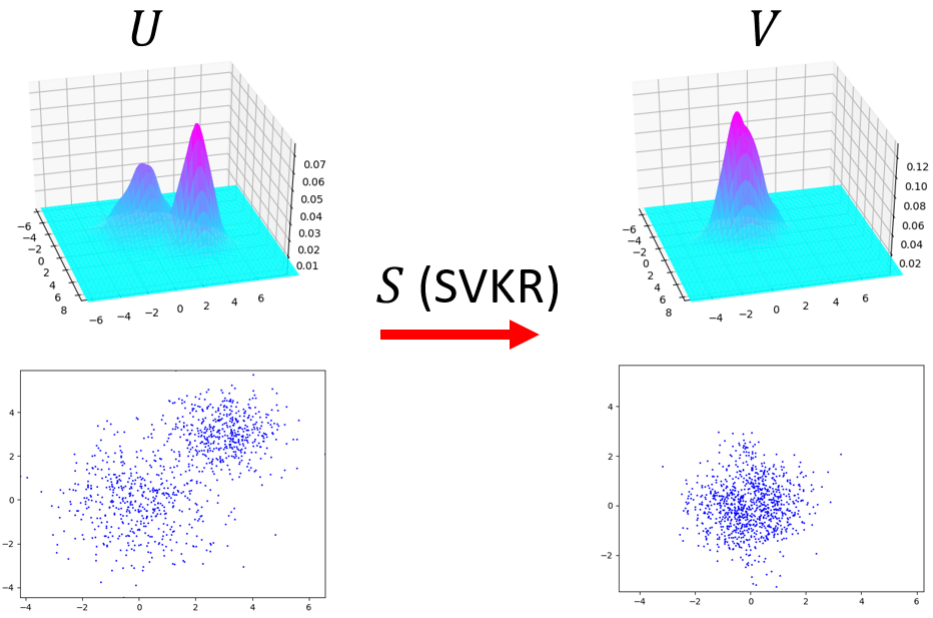


Figure 3.2. An example mapping computed to push a 2-dimensional bimodal distribution to a standard Gaussian distribution. Here, we notice a slight loss in the structure of V as a result of the decreased expressiveness of the parameterization

to help smooth out the samples (not adding the Gaussian noise essentially amounts to drawing from a discrete probability mass function as the pixels themselves are discrete bins, which is fully incompatible with our framework in a theoretical sense). This methodology of turning "images into samples" is not that out of the ordinary in the field of image processing with optimal transport maps, and is often used to investigate transport between 2 different images, so it will serve as an interesting stress test here for our algorithm.

Figure 3.3 is divided into 9 cells, one for each handwritten digit. The initial drawing of samples resulted in the top left and bottom left images in each cell showing samples distributed in the shape of each digit (bottom left of each cell), as well as a Kernel Density Estimate of these samples (top left of each cell). We then computed a Knothe-Rosenblatt map of order 8 to push each of these MNIST-digit distributions to a standard Gaussian, represented by the top right and bottom right images in each cell in the figure. While the pushes are seemingly headed in the right direction, there is clearly some degradation in the performance of the maps

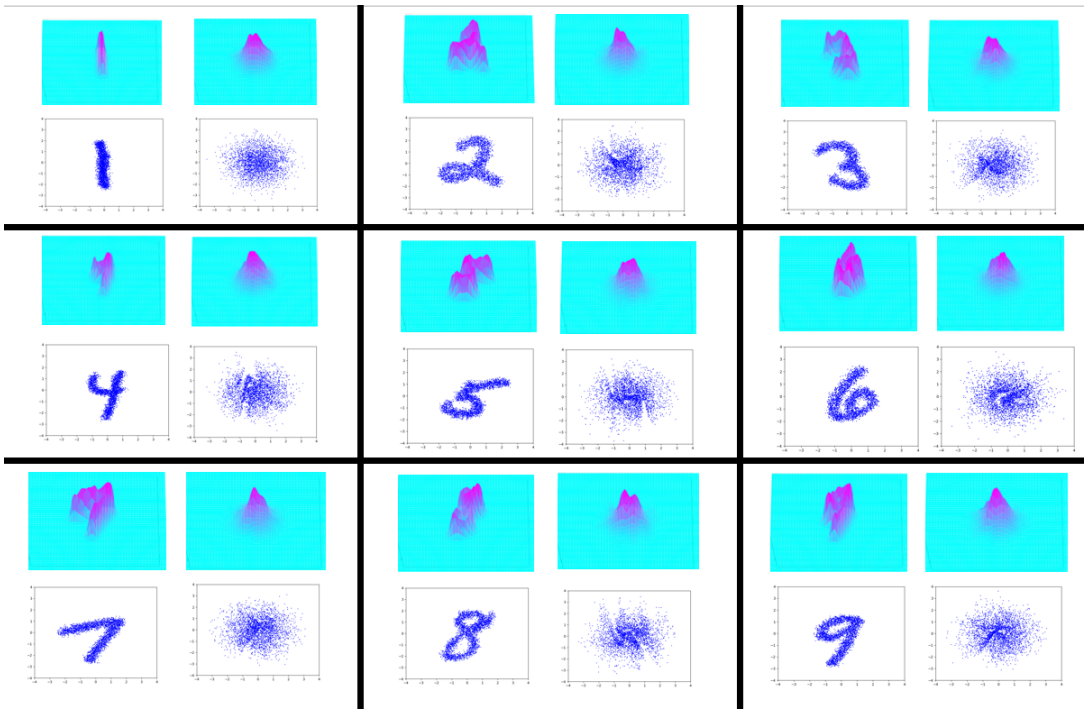


Figure 3.3. 9 examples of pushing a distribution represented by each of the MNIST handwritten digits in 2-dimensional space, to a 2-dimensional standard Gaussian distribution

when U is highly irregular. The push-to densities for most of these trials ended up having a mean of 0 and an essentially identity covariance, as desired. However, the overall structure of each V still falls somewhat short of the perfect symmetry of a standard Gaussian. Nevertheless, these transports did perform reasonably well, given the shape of U, especially in cases where U looked particularly out of the ordinary, such as the digit 3 or 5. In addition, because the original greyscale images are perfectly white text overlaid on perfectly black backgrounds, even adding the small amount of Gaussian noise to the initial sample draws may not be enough to circumvent some of the discontinuities at the border between where the digit was written on the black background, and the background itself. This may explain some of the discontinuous-like structure in V, such as where there are gaps of empty space in V (such as the the north-side of the push-to density for the digit 4, or the center of the push-to density for the digit 6), when it's rather supposed to be a perfectly smooth Gaussian distribution.

3.4 Sequential Mapping and Wasserstein Regularization

In this section, we introduce a scheme for using many individually computed maps in sequential composition to achieve an overall effect of a single mapping from U to V. By using a sequence of maps to transform U to V instead of a single map, one can theoretically *rely on models of lower complexity* to represent each map in the sequence, as although each map is, on its own, "weak" in its modeling complexity and ability to induce large changes in the distribution space, the combined action of many such maps together can potentially successfully transform samples as desired. This is especially attractive for model structures that increase exponentially in complexity with problem size, such as with the fully-expressive polynomial map described in previous sections. However, even in the case of the Single Variable Knothe-Rosenblatt maps where $K = DO + 1$, the length of the basis can still get quite high when in particularly high dimension, or when using extremely high order polynomials.

Moving forward, we first take a brief look at a non-equilibrium thermodynamics interpretation of this methodology to further justify the use of such a strategy. We believe the discussion pertaining to the physics of the problem is a very intriguing concept to think about in terms of linking the worlds of thermodynamics with optimal transport, and so we try to expand on that relationship in the following section. Subsequently, we then derive a slightly different ADMM problem to implement it using the well-studied Wasserstein distance as a compositional "regularization term" that will ultimately allow us to use lower-complexity maps in sequence to approximate a total map that achieves a more powerful transformation from U to V overall. The use of the word "regularization" will become more clear as our discussion continues.

3.4.1 Non-Equilibrium Thermodynamics and Sequential Evolution of Distributions

One approach to interpreting sequential composition of maps and justify its usage is to borrow ideas from statistical physics. Assume we would like to build a composition of maps in the following way:

$$S(x) = S_M \circ S_{M-1} \circ \ldots S_2 \circ S_1(x), \quad x \sim U \tag{3.64}$$

$$\text{s.t. } S\#U = V \tag{3.65}$$

where in this notation, the subscript of a map S_m represents the position of the map in the sequence.

Here, the "push-to" density v can be interpreted as an equilibrium density of particles in a system, which we can term ρ_∞, and every m^{th} map in the sequence can be considered one "step" of evolution of a distribution that has yet to reach equilibrium, starting with an out-of-equilibrium density at $m = 0$ of u. As such, for this section, we can consider the push-from density u as the non-equilibrium density at time-step 0, termed ρ_0. It is worth noting here that the ρ in this section is *not* the same as the ρ used in previous sections to denote the ADMM penalty parameter. But we use ρ here to represent densities to stay consistent with the notation of Jordan, Kinderlehrer and Otto's work in [38], where much of this discussion draws inspiration from.

Since ρ_∞ is an equilibrium density, it can be written as a Gibbs distribution (with temperature equal to 1 for simplicity): $v(x) = \rho_\infty(x) = Z^{-1}\exp(-\Psi(x))$. For example, if V is a standard Gaussian, then $\Psi(x) = \frac{1}{2}x^2$. Assuming the particles obey the Langevin equation, it is well known that the evolution of the particle density as a function of time, denoted $\rho_t : t \geq 0$, obeys the Fokker-Planck equation. It was shown in [38] that the trajectory of $(\rho_t : t \geq 0)$ can

be interpreted from variational principles. Specifically :

Theorem 3.1 ([38] Theorem 5.1) *Define $\rho_0 = u$, and $\rho_\infty = v$, and assume that $D(\rho_0||\rho_\infty) < \infty$. For any $h > 0$, consider the following minimization problem:*

$$F_1(\rho) \triangleq \frac{1}{2} W_2(\rho_{k-1}, \rho) + hD(\rho||\rho_\infty) \tag{3.66}$$

$$\rho_k \triangleq \underset{\rho \in \mathsf{P}(\mathsf{W})}{\arg\min}\, F_1(\rho) \tag{3.67}$$

where h serves as a hyperparameter. As $h \downarrow 0$, the piecewise constant interpolation which equals ρ_k for $t \in [kh, (k+1)h]$ converges weakly in $L_1(\mathbb{R}^D)$ to $(\rho_t : t \geq 0)$, the solution to the Fokker-Planck equation.

In the event that v is log-concave, this also has implications for the exponential convergence to equilibrium with this statistical physics perspective:

Theorem 3.2 (Bakry & Émery [6]) *If v is uniform log-concave, namely:*

$$\nabla^2 \Psi(x) \succeq \lambda I$$

for some $\lambda > 0$ with I the $D \times D$ identity matrix, then:

$$D(\rho_t||\rho_\infty) \leq e^{-2\lambda t} D(\rho_0||\rho_\infty)$$

We now note that for $h > 0$, 3.66 encodes a sequence $(\rho_m : m \geq 0)$ of densities which evolve towards $\rho_\infty = v$. As such, from Theorem 1.1, there exists an $S_1 \in \mathfrak{D}_+$ for which $S_1 \# \rho_0 = \rho_1$, and more generally, for any $m \geq 0$, there exists $S_m \in \mathfrak{D}_+$ for which $S_m \# \rho_{m-1} = \rho_m$. As 3.66 operates on the evolution of the probability densities themselves, our goal now is to explicitly recast the theoretical result from 3.66 in terms of our optimal transport maps S_m instead. With that, we have the following theorem and proof:

Theorem 3.3 (Sequence of Maps) *Define* $F_2 : \mathfrak{D}_+ \to \mathbb{R}$ *as:*

$$F_2(S) \triangleq \frac{1}{2}\mathbb{E}_{\rho_{m-1}}[||X - S(X)||^2] + hD(\rho_{m-1}||\tilde{u}(\cdot;S))$$
$$S_k \triangleq \arg\min_{S\in\mathfrak{D}_+} F_2(S) \tag{3.68}$$

Then $F_1(\rho_k) = F_2(S_k)$ *and* $S_k\#\rho_{m-1} = \rho_m$

Proof We first recall from Section 2.1 that $\tilde{u}(\cdot;S)$ represents the fact that a distribution $\tilde{u}$ is induced by a candidate mapping S, and a given push-to distribution v, which in this case is ρ_∞. Combined with the invariance of relative entropy under an invertible transformation, for a map $S \in \mathfrak{D}_+$, the following are equivalent:

$$D(\rho_{m-1}||\tilde{u}(\cdot;S)) = D(\rho_{m-1}||S^{-1}\#\rho_\infty) = D(S\#\rho_{m-1}||\rho_\infty) \tag{3.69}$$

As such, we can also define the following F_3:

$$F_3(S) \triangleq \frac{1}{2}\mathbb{E}_{\rho_{m-1}}[||X - S(X)||^2] + hD(S\#\rho_{m-1}||\rho_\infty) \tag{3.70}$$

and furthermore, $F_3(S) = F_2(S)$.

From Corollary 1.1, $W_2(\rho_{m-1}, S\#\rho_{m-1}) \leq \mathbb{E}_{\rho_{m-1}}[(X - S(X))^2]$ for $S \in \mathfrak{D}_+$. Also, since the relative entropy terms of F_3 and F_1 are equal, it follows that $F_3(S) \geq F_1(S\#\rho_{m-1})$ for $S \in \mathfrak{D}_+$. And finally, recalling that Corollary 1.1 tells us that $S_m \in \mathfrak{D}_+$ such that $S_m\#\rho_{m-1} = \rho_m$ exists, we can conclude that

$$\mathbb{E}_{\rho_{m-1}}[||X - S_m(X)||^2] = W_2(\rho_{m-1}, \rho_m) \tag{3.71}$$

and so $F_3(S_k) = F_2(S_k) = F_1(\rho_k)$. □

As a result, we can now show how a composition of maps underlies how a sample from

$U = \rho_0$ can give rise to a sample from ρ_m:

$$\rho_m = S_m \# \rho_{m-1} = S_m \circ S_{m-1} \# \rho_{m-2} = S_m \circ \cdots \circ S_1 \# \rho_0 \tag{3.72}$$

Moreover, since as $h \downarrow 0$, S_m starts to approach the identity map and $\rho_m \simeq \rho_{m-1}$. Thus for small $h > 0$, each S_m should be estimated with reasonable accuracy using lower-order map parameterizations. That is, S can be described as the composition of M maps as:

$$S(x) = S_M \circ \ldots \circ S_2 \circ S_1(x) \tag{3.73}$$

where $x \sim U$, such that each S_m is of relative low-order in the polynomial chaos expansion (assuming we're using polynomial representations of the map; in a more general sense, a better phrasing might be that the sequential maps can be "lower-complexity" by this argument).

Note that (3.68) as written above involves the expectation with respect to ρ_{m-1}. Since our scheme operates sequentially, we have already estimated $S_1, S_2, \ldots, S_{m-1}$, and generate approximate i.i.d samples from ρ_{m-1} by first generating $X_i \sim \rho_0 = u$, $i = 1, \ldots, N$, and constructing

$$Z_i = S_{m-1} \circ \cdots \circ S_1(X_i) \tag{3.74}$$

Recalling Equation (2.4) from our original KL-Divergence based optimization problem, and its equivalence to Equation (2.8) in an optimization setting, we can finally arrive at the primary formulation we will be working with in a sequential mapping setting. We can apply the Wasserstein distance to the objective in exactly the same way as in Equation (3.66) to give us:

$$\min_{S \in \mathfrak{D}_+} \frac{1}{N} \sum_{i=1}^{N} \left[\theta ||Z_i - S(Z_i)||^2 - \log \tilde{u}(Z_i; S) \right] \tag{3.75}$$

where the sum is approximating an expectation with respect to ρ_{m-1} instead of (necessarily) $\rho_0 = U$, and $\theta = \frac{1}{2h}$ can be thought of as an inverse "step-size" in the following

way: if we are computing the m^{th} map in a sequential chain, we can generate independent samples from ρ_{m-1} using (3.74), and solve problem (3.75) to find a map S_m^* that pushes ρ_{m-1} forward closer to $\rho_\infty = V$. The extent to how "far" the S_m^* manages to push ρ_{m-1} towards ρ_∞ is dictated by the parameter θ, where as θ gets larger, S_m^* begins to approach the identity map. As $\theta \downarrow 0$, the problem can be interpreted as taking more aggressive steps to map ρ_{m-1} to ρ_∞, and indeed, when $\theta = 0$, we have exactly Equation (2.8) which intends to find a single mapping to push from U to V; as such, the advantage to having larger θ is it *regularizes* the extent of the transport at each step, and so theoretically, lower complexity maps will suffice to perform the individual, smaller, steps in the compositional transformation. Hence, as the number of compositions M in (3.73) increases, ρ_M approaches ρ_∞. Even moreover, when v is uniform log-concave, then by Theorem 3.2, this greedy, sequential approach still guarantees exponential convergence.

3.4.2 Putting it all together: Sequential Knothe-Rosenblatt Maps

To combine the analysis with our Knothe-Rosenblatt optimization problem, Equation (3.38), we first recall that in the KR case:

$$\min_{\tilde{S}\in\mathfrak{D}_+} D(U||\tilde{U}(\cdot;\tilde{S}) \tag{3.76}$$

$$= \min_{\tilde{S}in\mathfrak{D}_+} \mathbb{E}_U[-\log \tilde{u}(X;\tilde{S})] \tag{3.77}$$

$$= \min_{\tilde{S}\in\mathfrak{D}_+} \mathbb{E}_U[-\log v(\tilde{S}(X)) - \log\det J_{\tilde{S}}(X)] \tag{3.78}$$

$$= \min_{\tilde{S}\in\mathfrak{D}_+} \mathbb{E}_U[-\log v(\tilde{S}(X)) - \sum_{d=1}^{D} \log \partial_d \tilde{S}^d(X)] \tag{3.79}$$

$$\approx \min_{\tilde{S}\in\mathfrak{D}_+} \frac{1}{N}\sum_{i=1}^{N}[-\log v(\tilde{S}(X_i)) - \sum_{d=1}^{D} \log \partial_d \tilde{S}^d(X_i))] \tag{3.80}$$

where (3.76) is the original KL-Divergence based optimal transportation problem that ultimately equates to both (3.77) and (3.78) as shown in Section 2.1 (just reproducing here for

convenience), (3.79) follows from Equation (3.36) and the Knothe-Rosenblatt property of the map, and (3.80) is a Monte Carlo approximation of the expectation.

Then, by combining with (3.75), we arrive at our 2-Wasserstein regularized Knothe-Rosenblatt transport problem:

$$S^* = \arg\min_{\tilde{S}} \frac{1}{N}\sum_{i=1}^{N} \theta||X_i - \tilde{S}(X_i)||_2^2 - \log v(\tilde{S}(X_i)) - \sum_{d=1}^{D} \log \partial_d \tilde{S}^d(X_i) \tag{3.81}$$
$$\text{s.t. } \partial_d \tilde{S}^d(X_i) > 0, \quad d = 1,2,\ldots,D \quad i = 1,2,\ldots,N$$

where $\mathfrak{D}_+^{KR}$ is the set of all monotonic diffeomorphisms with the Knothe-Rosenblatt property. Though each map must be calculated sequentially after the previous one (so as to generate samples from ρ_{m-1} at every sequence step), each map in the sequence can still be computed using the distributed framework described in Section 3.3.1. In Figure 3.4, we show an example of a sequential mapping in action, with intermediate steps in the sequence pictured to emphasize the gradual nature of the push from U, a bimodal 2-dimensional distribution, to V, a standard Gaussian distribution. Each map in the sequence was a Full Knothe-Rosenblatt map of order 8, with $\theta = 5$. In this example, the first, tenth, twentieth, and thirtieth mappings in the sequence are represented using both a Kernel Density Estimate of the transported samples at that sequential mapping stage (top row), as well as using the transported samples themselves (bottom row).

Note that this scheme is also compatible with the non-KR version of the formulation from Section 3.2, with the appropriate reversion to the log det term in the objective function. But we specifically frame the problem from the perspective of the KR formulation first and foremost, as this is what we actually used in the applications involving sequential mappings that will be discussed in Chapter 5.

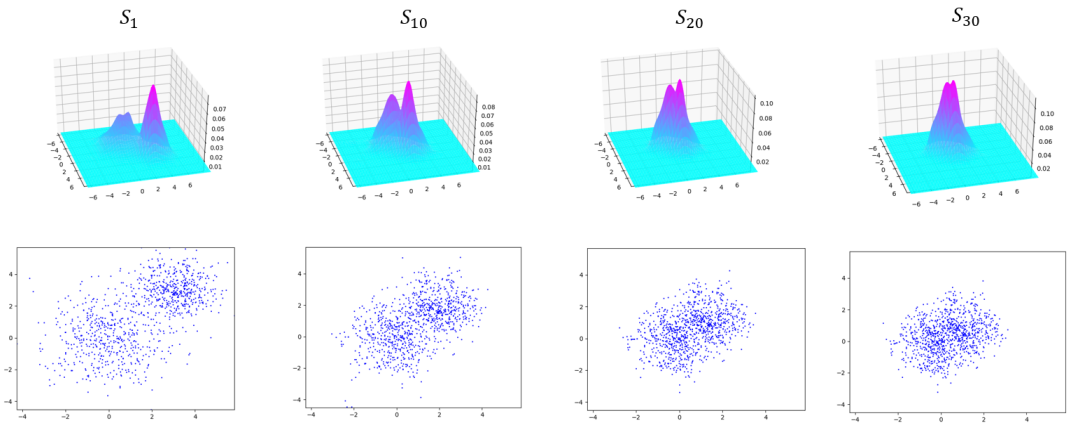


Figure 3.4. An example of a sequential mapping to push a bimodal distribution to a standard Gaussian distribution. S_1, S_{10}, S_{20}, and S_{30} are represented here, and as we move forward in the sequence, we see the push gradually beginning to approach V.

3.4.3 Change to Closed-Form Updates

In truth, the one and only change to the closed-form updates for the KR-ADMM framework is actually only to the update of the consensus variable, B. This manifests from the fact that B is the only ADMM variable involved in the 2-Wasserstein term in the objective function in (3.81), via $S(X_i)$. Because of how simple this change is, and for the sake of conciseness, we will simply only show two things in this subsection: we will show the fully-penalized Lagrangian of the updated problem, followed by the updated closed-form solution of B (i.e. omitting the more in-depth derivations we did for the original problem). We will highlight the only changes made to the original framework in red in the following equations.

The fully-penalized Lagrangian of the 2-Wasserstein regularized problem is:

$$
\begin{aligned}
&L_{\rho,\theta}(\tilde{B}_i, Z_i^d, Y_i^d, p_i, B; \gamma_i, \alpha_i, \beta_i^d, \lambda_i^d) \\
&= \frac{1}{N}\sum_{i=1}^{N} \theta||B\Phi_i - X_i||_2^2 - \log v(p_i) + \gamma_i^T(p_i - B\Phi_i) + \mathbf{tr}(\alpha_i^T(\tilde{B}_i - B)) \\
&+ \frac{\rho}{2}||\tilde{B}_i - B||_2^2 + \frac{\rho}{2}||B\Phi_i - p_i||_2^2 \\
&+ \sum_{d=1}^{D} -\log Z_i^d + \beta_i^d(Z_i^d - Y_i^d \mathbf{1}_d) + \lambda_i^{dT}(Y_i^d - B\Phi_i^d) \\
&+ \frac{\rho}{2}(Y_i^d \mathbf{1}_d - Z_i^d)^2 + \frac{\rho}{2}||B\Phi_i^d - Y_i^d||_2^2
\end{aligned} \tag{3.82}
$$

where $X_i,\ i = 1,\ldots,N$ are samples drawn from $u = \rho_{m-1}$. And the updated B update is:

$$
\begin{aligned}
B &= M \cdot L^{-1} \\
M &\triangleq \frac{1}{N}\sum_{i=1}^{N} \rho\tilde{B}_i^{(k)} + \rho p_i^{(k)}\Phi_i^T + 2\theta X_i\Phi_i^T + \gamma_i^{(k)}\Phi_i^T + \alpha_i^{(k)T} \\
&\quad + \sum_{d=1}^{D} \rho Y_i^{d(k)}\Phi_i^{dT} + \lambda_i^{d(k)}\Phi_i^{dT} \\
L &\triangleq \rho\left(I + \frac{1}{N}\sum_{i=1}^{N}\Phi_i\Phi_i^T + \frac{2\theta}{\rho}\Phi_i\Phi_i^T + \sum_{d=1}^{D}\Phi_i^d\Phi_i^{dT}\right)
\end{aligned} \tag{3.83}
$$

3.5 Acknowledgements

This chapter featured reprints of much of the material from “A distributed framework for the construction of transport maps”, Mesa, Diego A.; Tantiongloc, Justin; Mendoza, Marcela; Coleman, Todd P., which has been submitted to Neural Computation in 2018, [52], of which the dissertation author was a primary co-investigator.

Chapter 4

ADMM Implementation Details

This chapter will go through some of the implementation details of the Knothe-Rosenblatt-specific formulation from Section 3.3. Especially for applications in higher dimensions and with a larger number of input samples for computing the map, using the Single Variable Knothe-Rosenblatt parameterization of the mapping is often the most feasible way to keep problem complexity to tractable levels, both in terms of computation time, as well as memory consumption. As a result of this observation, we will be focusing specifically on the Knothe-Rosenblatt formulation in terms of our discussion on implementation details.

4.1 Using ADMM To Parallelize Over Threads

The ADMM framework allows for easy parallelization in the case of problems that can be reformulated as a global consensus problem, such as what we've done in the previous sections. If we consider the Knothe-Rosenblatt optimization problem, Equation (3.39), the problem is very clearly separable across N, as every n^{th} optimization variable is only dependent on the n^{th}-indexed version of every other variable, with the exception of the consensus variable B, which is globally shared amongst all parallel problems.

This leads to the natural parallelization scheme depicted in Figure 4.1. Here, we begin with a "distribute" phase where a master thread distributes the current value of $B^{(k)}$ to all N threads for use in associated variable updates, with all variables initialized randomly at the

beginning of the ADMM process. This is the one phase of the process that is not parallelized via our ADMM formulation (i.e. we still take advantage of parallelizable operations as far as running linear algebra/other mathematical operations on the GPU whenever we can, but the B update has no parallelizability from the perspective of ADMM). Each n^{th} thread then iterates over the rest of their associated n^{th} optimization variables, including their own version of the mapping coefficients, $\tilde{B}_i$. After the iteration of variable updates is over, we enter a "reduction" phase, where the results from each thread are consolidated by the master thread to compute $B^{(k+1)}$. In terms of computation time, the B-update is definitely the bottleneck, as it involves what amounts to an ND-long loop of outer products in the Knothe-Rosenblatt case, at every ADMM iteration, and cannot be parallelized with respect to any of the ADMM variables. Note that in this figure, dotted boxes labeled $d = 1 \dots D$ represent operations that must be performed D times, for D-many variables. However, each of these D variables can be considered its own ADMM variable, and thus can be updated sequentially as well.

However, upon closer examination of (3.53), (3.54) and (3.55), or (3.83) in the regularized case, we see that L is actually a static component of the computation - that is, it does not depend on any of the iterative ADMM variables at all. As such, given the initial input samples $X_i,\ i = 1, \dots, N$, we can pre-compute this component of the update before ADMM even begins to save on computation time.

4.2 Polynomial Representation

As mentioned in Section 2.2, one can select which polynomial basis they would like to use based on the structure of U [81, 67], in the sense that certain polynomials result in different convergence rates of the approximation. For example, if U is a Gaussian distribution, it is ideal to use the Hermite polynomials to parameterize the map. Similarly, if U is the Gamma distribution, it is ideal to use the Laguerre polynomials [81].

However in practice, when U has arbitrary structure, the Hermite polynomials can

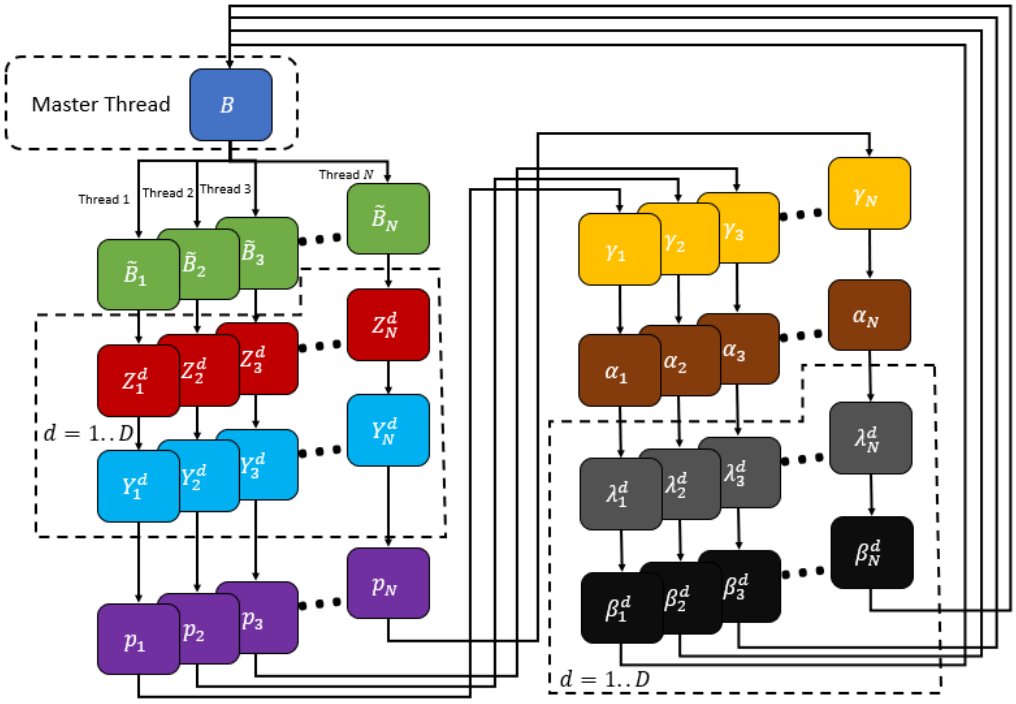


Figure 4.1. A high level graphic of the ADMM process with respect to our optimal transportation framework. The key thing to note is that all ADMM variable updates can be parallelized across N, and the only update that requires all other variables is the B-update.

provably approximate any mapping, as long as K is large enough. Therefore, for all of the applications we will discuss in Chapter 5, we do indeed use the Hermite polynomials as the map parameterization.

In terms of implementation, the simplest way we've found to represent the index sets $\mathscr{J}$, $\mathscr{J}^{KR}$ and $\mathscr{J}^{KRSV}$ is by maintaining a simple $D \times K$ matrix of univariate indices, as this leads to a very natural way to enforce the lower-triangular constraint for Knothe-Rosenblatt maps. Here, we'll give a few concrete examples for clarification in practical use-cases and implementation.

In the case of the fully-expressive map, recall that the index set is:

$$\mathscr{J} = \left\{ \mathbf{j} \in \mathbb{N}^D : \sum_{i=1}^{D} j_i \leq O \right\}$$

where O is the maximum allowed order of the polynomial basis. For example, when $D = O = 3$, the resulting index set matrix will have the following form:

$$\mathscr{J} = \begin{bmatrix} 0 & 1 & 2 & 3 & 0 & 0 & 0 & 1 & 1 & 2 & 0 & 0 & 0 & 1 & 1 & 2 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 3 & 1 & 2 & 1 & 0 & 1 & 2 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 3 \end{bmatrix}$$

where every k^{th} column represents $\mathbf{j}_k$, one D-long multi-index for a single multivariate polynomial basis term, $\phi_{\mathbf{j}_k}$. In each k^{th} column, each element represents the order of the univariate polynomial in the tensor product associated with $\phi_{\mathbf{j}_k}$. So when $\mathbf{j}_k = [\mathbf{j}_k^1, \mathbf{j}_k^2, \mathbf{j}_k^3]$, and $x_i = [x_i^1, x_i^2, x_i^3]$, then $\phi_{\mathbf{j}_k}(x_i) = \prod_{d=1}^{D} \psi_{\mathbf{j}_k^d}(x_i^d)$, where ψ_n is the n^{th} univariate polynomial in the polynomial series chosen (e.g. Hermite, Laguerre, etc.). We can then build each basis vector $\Phi(x_i)$ for some input vector x_i by iterating over the columns of this matrix, and building $[\phi_{\mathbf{j}_1}(x_i), \phi_{\mathbf{j}_2}(x_i), \cdots, \phi_{\mathbf{j}_K}(x_i)]$ accordingly for use in our optimization problem.

In the event that we want to switch to a different map parameterization, all we need to

do is change the members of this matrix. Recall the definition of $\mathscr{J}_d^{KR}$:

$$\mathscr{J}_d^{KR} = \left\{\mathbf{j} \in \mathbb{N}^D : \sum_{i=1}^{D} j_i \leq O \wedge j_i = 0, \forall i > d\right\}, d = 1, \ldots, D$$

In this case, the size of the set $K_d \triangleq |\mathscr{J}_d^{KR}|$ becomes dependent on the component of the mapping. Revisiting our previous example with $D = O = 3$, we have the following 3 multi-index sets:

$$\begin{aligned}
\mathscr{J}_1^{KR} &= \left\{\mathbf{j} \in \mathbb{N}^3 : \sum_{i=1}^{3} j_i \leq O \wedge j_2 = j_3 = 0\right\} \\
&= \begin{bmatrix} 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\
\mathscr{J}_2^{KR} &= \left\{\mathbf{j} \in \mathbb{N}^3 : \sum_{i=1}^{3} j_i \leq O \wedge j_3 = 0\right\} \\
&= \begin{bmatrix} 0 & 1 & 2 & 3 & 0 & 0 & 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 0 & 1 & 2 & 3 & 1 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\
\mathscr{J}_3^{KR} &= \left\{\mathbf{j} \in \mathbb{N}^3 : \sum_{i=1}^{3} j_i \leq O\right\} \\
&= \begin{bmatrix} 0 & 1 & 2 & 3 & 0 & 0 & 0 & 1 & 1 & 2 & 0 & 0 & 0 & 1 & 1 & 2 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 3 & 1 & 2 & 1 & 0 & 1 & 2 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 3 \end{bmatrix}
\end{aligned} \tag{4.1}$$

To implement a lower-triangular map, we can simply construct Φ according to the full multi-index set ordering of $\mathscr{J}_D^{KR}$, and constraining the coefficient matrix B to have zeros embedded with the following structure:

Definition 4.1 (Lower-Triangular Coefficient Matrix) . *A coefficient matrix $B \in \mathbb{R}^{D \times K}$ corresponds to a lower-triangular, or Knothe-Rosenblatt, transport map if it can be expressed as:*

$$B = \begin{bmatrix} \mathbf{w}_1^T & 0 & 0 & 0 & 0 & 0 & 0 \\ & \ldots & \mathbf{w}_d^T & \ldots & 0 & 0 & 0 \\ & & \ldots & \mathbf{w}_D^T & \ldots & & \end{bmatrix} \tag{4.2}$$

where each $\mathbf{w}_d \in \mathbb{R}^{K_d}$

In other words, to create a lower-triangular mapping, we just ensure that every d^{th} row of the coefficient matrix B has, at most, K_d non-zero coefficients at the beginning of each row. This makes sense as long as the multi-index matrix is constructed such that each $\mathscr{J}_d^{KR}$ constitutes the left $D \times K_d$-sized submatrix of $\mathscr{J}_{d+1}^{KR}$, as depicted in (4.1); i.e. notice how the columns of $\mathscr{J}_1^{KR}$ are also the first $K_1 = 4$ columns of $\mathscr{J}_2^{KR}$, and so on. This implies that the first K_1 members of Φ will only be a function of x_1, the first K_2 members of Φ will only be a function of x_i^1 and x_i^2, etc. The following matrix representation should make things more clear. This is an expanded view of the mapping function $B\Phi_i = S(x_i)$:

$$\underbrace{\begin{bmatrix} b_{1,1} & b_{1,2} & \dots & b_{1,K_1} & \dots & 0 & 0 & 0 \\ b_{2,1} & b_{2,2} & \dots & \dots & b_{2,K_2} & \dots & 0 & 0 \\ \vdots & & & & & & & \\ b_{D,1} & b_{D,2} & \dots & \dots & \dots & \dots & \dots & b_{D,K_D} \end{bmatrix}}_{B} \underbrace{\begin{bmatrix} | \\ \Phi(x_i^1) \\ | \\ | \\ \Phi(x_i^1, x_i^2) \\ | \\ \vdots \\ | \\ \Phi(x_i^1, \dots, x_i^D) \\ | \end{bmatrix}}_{\Phi_i}$$

$$= \begin{bmatrix} S^1(x_i^1) \\ S^2(x_i^1, x_i^2) \\ \vdots \\ S^D(x_i^1, \dots x_i^D) \end{bmatrix} \tag{4.3}$$

To fulfill our KR assumption, we assume that Φ_i is a column vector of the polynomial bases

evaluated at x_i, ordered according to how many components of x_i the bases are a function of. I.e., if $K_d = |\mathscr{J}_d^{KR}|$, then $\Phi(x_i^1)$ are the first K_1 basis functions that are only a function of x_1, $\Phi(x_i^1, x_i^2)$ are the $K_2 - K_1$ basis functions that are only a function of x_1 and x_2, and so on. As such, as only the first K_d elements of every d^{th} row of B are potentially non-zero, the map should have the appropriate Knothe-Rosenblatt structure by construction.

In the case of the Single Variable Knothe-Rosenblatt map, the index set becomes the following subset of $\mathscr{J}_d^{KR}$ for all d:

$$\mathscr{J}_d^{KRSV} = \left\{\mathbf{j} \in \mathbb{N}^D : \sum_{i=1}^{D} j_i \leq O \wedge j_i j_l = 0, \forall i \neq l \wedge j_i = 0, \forall i > d\right\}, d = 1, \ldots, D$$

Creating an index matrix that enforces this property requires simply removing all columns from $\mathscr{J}_d^{KR}$ that have more than one non-zero element in them. So for $D = O = 3$:

$$\begin{aligned}
\mathscr{J}_1^{KR} &= \left\{\mathbf{j} \in \mathbb{N}^3 : \sum_{i=1}^{3} j_i \leq O \wedge j_2 = j_3 = 0 \wedge j_i j_l = 0, \forall i \neq l\right\} \\
&= \begin{bmatrix} 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\
\mathscr{J}_2^{KR} &= \left\{\mathbf{j} \in \mathbb{N}^3 : \sum_{i=1}^{3} j_i \leq O \wedge j_3 = 0 \wedge j_i j_l = 0, \forall i \neq l\right\} \\
&= \begin{bmatrix} 0 & 1 & 2 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\
\mathscr{J}_3^{KR} &= \left\{\mathbf{j} \in \mathbb{N}^3 : \sum_{i=1}^{3} j_i \leq O \wedge j_i j_l = 0, \forall i \neq l\right\} \\
&= \begin{bmatrix} 0 & 1 & 2 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 3 \end{bmatrix}
\end{aligned} \tag{4.4}$$

In terms of implementation, the 0-embedding strategy from Definition 4.1 still applies, as long as the complete index set is constructed as $\mathscr{J}_D^{KRSV}$, and the number of non-zero coefficients in each d^{th} row is $|\mathscr{J}_d^{KRSV}|$ instead.

4.3 Inverting Knothe-Rosenblatt Maps

In this section, we will go into more detail about a means to invert a Knothe-Rosenblatt mapping using this polynomial representation that was previously mentioned in [49] and [52]. Computing the inverse map becomes fairly straightforward given the above methodology of representing B and Φ_i.

If we want to invert a sample $S(x_i)$, this defines a system of equations that can be solved row-by-row for each component of the mapping $S(x_i^d)$ in the form of a polynomial root-finding problem for each row. or example, we first solve for x_i^1, the solution of which we can call x_i^{1*} by finding the (single variable) root of:

$$\begin{bmatrix} b_{11} & b_{12} & \dots & b_{1(K_1)} \end{bmatrix} \begin{bmatrix} | \\ \Phi(x_i^1) \\ | \end{bmatrix} = S(x_i^1) \tag{4.5}$$

Subsequently, we can solve for x_i^{2*} plugging in x_i^{1*} into the second equation:

$$\begin{bmatrix} b_{21} & b_{22} & \dots & \dots & b_{2(K_2)} \end{bmatrix} \begin{bmatrix} | \\ \Phi(x_i^{1*}) \\ | \\ | \\ \Phi(x_i^{1*}, x_i^2) \\ | \end{bmatrix} = S(x_i^2) \tag{4.6}$$

and so on. This is only possible because of the Knothe-Rosenblatt property, and the "layer-by-layer" increasing chain of dependencies on components of x_i as you iterate over the dimension of the mapping. Ultimately, this results in D-many single variable root-finding problems per sample to invert, and the order of the polynomial that must be solved for will be equal to the

order of the polynomial chosen to represent the basis.

4.4 Running on GPUs

Our CUDA implementation of this framework allows us to run our algorithm on Nvidia GPUs, and to take advantage of the very high parallelizability of these devices. The original prototype implementation of our ADMM formulation was designed for a single CPU thread, and Figure 4.2 compares the wall-clock performance *in seconds* between the single-CPU-thread implementation of the ADMM problem (i.e. no parallelization, and all individual N-many ADMM variables are updated sequentially) and the single-GPU implementation of the ADMM problem. The figure compares 4 cases, of varying problem sizes (N) and varying dimension (d), for the same toy problem used before of pushing a bimodal distribution to a standard Gaussian, extrapolated to higher dimension, of course. For each case, full Knothe-Rosenblatt maps of various polynomial orders ranging from 2 to 5 were compared (the absence of entries for certain orders is due to lack of memory, once again emphasizing the exponential increase of basis terms when using the full KR mapping), with computation time being determined by convergence of the iterative change of the dual variables to a set threshold. The key thing to note in this figure is the fact that the speedup associated with the parallelized GPU implementation is quite dramatic in every single case, even for $d = 10$, which would very much be applicable to many image processing applications.

However, we obviously wish to accommodate even higher-dimensional problems if possible. To this end, there are two possible solutions:

1. Use the Single Variable Knothe-Rosenblatt mapping

2. Use more GPUs

Indeed, our GPU implementation is also designed to accommodate multiple GPUs running on the same host, and with research support from Amazon Web Services, we were able to

N = 2000, d=4	order 2	order 3	order 4	order 5
CPU	21.87	26.54	23.24	29.17
GPU	**0.204**	**0.41**	**1.12**	**1.17**
N=3000, d=4				
CPU	34.75	40.05	42.16	46.05
GPU	**0.278**	**0.569**	**1.45**	**2.007**
N=2000, d=7				
CPU	37.345	54.78	71.95	
GPU	**0.737**	**2.819**	**7.62**	
N=1000, d=10				
CPU	21.685	42.95		
GPU	**0.85**	**4.38**		

Figure 4.2. Comparison of wall-clock computation time in seconds for various problems using the CPU implementation and GPU implementation on a single GPU

benchmark many larger problem sizes with respect to the Single Variable Knothe-Rosenblatt mapping, *and* with respect to running multiple GPUs. Figure 4.3 shows the results of these benchmark tests. We performed the same transport problem, from a bimodal distribution to a standard Gaussian, for dimensions 5,10,20,50,100,150 and 200, with a constant number of samples of $N = 1000$. Each of the maps in this figure was also a composition of 10 sequential Single Variable Knothe-Rosenblatt maps of order 2, and $\theta = 3$. Convergence was once again determined by monitoring the change in dual variables. The orange curve labeled "1 GPU" was obtained using a single Nvidia GTX 1080ti GPU, and the blue "AWS" curve corresponds to the performance using an AWS instance equipped with 8 Nvidia Tesla V100 GPUs. The 1-GPU curve terminates prematurely as the 1080ti ran out of memory for this specific setup around a problem size of $D = 230$, whereas the 8-GPU can go well beyond that (the 784-dimensional MNIST problem discussed in Section 5.3 uses this exact same AWS instance).

The trending of the curves shows that, as expected, as problem dimension increases,

a multi-GPU system will continue to maintain reasonable computation times, at least when compared to the single-GPU system. Furthermore, even if the orange curve were extrapolated further (beyond the memory limits of the 1080ti), the trending pattern of the two curves indeed suggests that the discrepancy of the computation time as dimension grows larger would only become more pronounced. In other words, the multi-GPU system seems to develop an increasingly large speedup multiplier with respect to the single-GPU system as dimension increases, as the blue curve is growing at a much smoother pace. It is worth noting that the multi-GPU curve is still experiencing exponential growth, however. This makes sense, as we are not only actually parallelizing the computation with respect to the number of samples, which remains a static size of 1000, but we are also taking advantage of parallelized linear algebra operations whenever we can, such as for all of the dot products and outer products being performed in our closed-form ADMM solutions. All of those operations require more parallel threads as problem size increases. As a result, the finite number of parallel threads the GPUs can accommodate does saturate faster as dimension increases, thus resulting in CUDA kernels having to queue up for execution after a certain problem size is reached.

But nevertheless, these results show a healthy way to scale our problems to relatively high dimensions, with computation times that are very much reasonable for their associated problem sizes. In terms of memory usage, while it may seem somewhat of a "brute-force" solution, allowing scalability across GPU's also provides a way to distribute samples and their corresponding ADMM variables across multiple devices allowing for the algorithm to continue to accommodate more memory-intensive problems as we add more devices to the host or cluster, all by virtue of the ADMM formulation.

4.5 Acknowledgements

This chapter featured reprints of much of the work from "A distributed framework for the construction of transport maps", Mesa, Diego A.; Tantiongloc, Justin; Mendoza, Marcela;

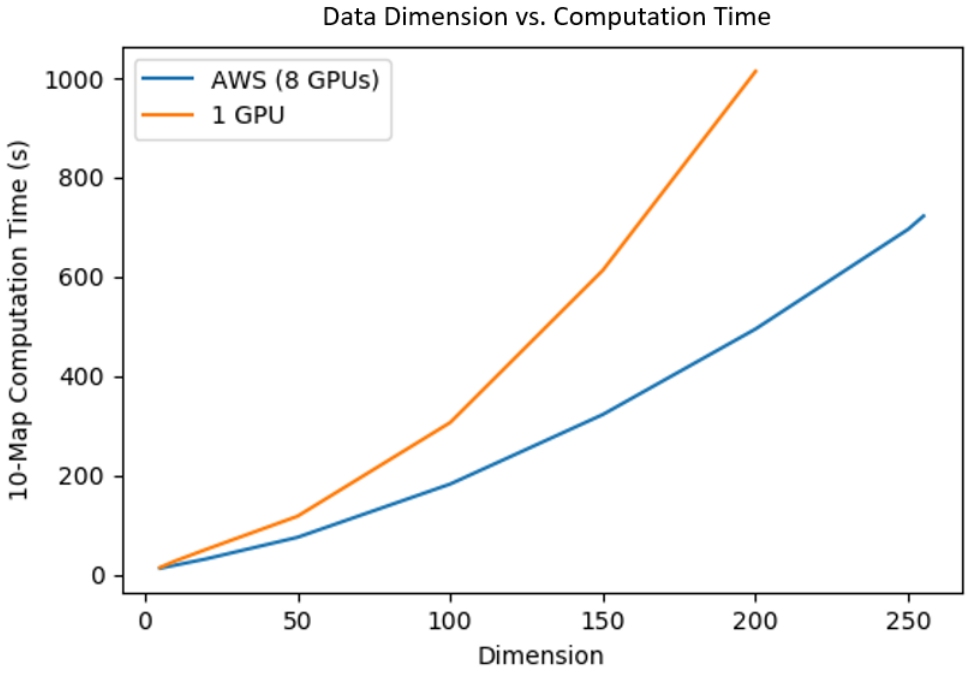


Figure 4.3. Comparison of wall-clock computation times in seconds vs. dimension on a single GPU system to an 8-GPU system on AWS

Coleman, Todd P., which has been submitted to Neural Computation in 2018, [52], of which the dissertation author was a primary co-investigator. We would also like to acknowledge Amazon Web Services for awarding us a full year's worth of research credits to use AWS's GPU compute nodes to carry out some of our more taxing evaluation benchmarks, including most of the applications in the next chapter.

Chapter 5

Applications

In this chapter, we will be taking a close look at some of the applications we have applied our optimal transportation framework to. These applications are summarized as follows: a multi-user brain-computer interface [73], a density estimation problem for the automated classification of sleep stages using multivariate EEG data [16, 52], and a demonstration of our ability to produce high-dimension maps via a generative model for MNIST handwritten digits [52]. In addition, we will showcase preliminary findings for an application of our framework to a Bayesian inference problem in the context of a preference-based deep reinforcement learning framework, that, while not yet completed, has shown extremely promising results that very much warrant additional research.

5.1 An Application to Posterior Matching and a Multi-User Brain-Computer Interface

This section is primarly a reprint of much of the material from "An information and control framework for optimizing user-compliant human computer interfaces" in the Proceedings of the IEEE, Tantiongloc, Justin; Mesa, Diego A.; Ma, Rui; Kim, Sanggyun; Alzate, Cristian H.; Camacho, Jaime J.; Manian Vidya; Coleman, Todd P., 2017, [73], with some simplifications and conceptual omissions made to better suit the context of this dissertation. For a complete version of the discussion as it pertains to team decision theory, and feedback-based

communication over a noisy channel, the reader is highly encouraged to read [73] in its entirety, as many of the theoretical aspects of the system are described in full within that work, but not necessarily here.

Using the optimal transportation framework described above applied to $\mathbb{R}^2$, we instantiate a novel, proof-of-concept brain-computer interface (BCI) where the goal is to allow 2 human users to specify a point on a geographic map to a computer learning agent using only brain signals measured with electroenchephalography (EEG). It is worth noting that this project was completed chronologically *before* the formulation of even the first ADMM problem described in [51], and as such, used a purely serial, CPU-based implementation of our optimal transportation problem.

The BCI we will be discussing in this section highlights two very important concepts that lie at the heart of its design:

- By using optimal transport, we can efficiently structure the communication between the human users and the computer system over the inherently noisy EEG communication channel in order to maximize the rate of information communication
- We also adhere to design principles pertaining to human-computer interfaces that aim to make the task for the users as conceptually simple as possible by abstracting away details of the underlying system from the users whenever possible

5.1.1 Motor Imagery

We will be leveraging motor imagery as the signaling mechanism between the human and computer agent in this project, as it is a particularly well-studied neurological phenomenon that is provably easy to identify and classify using EEG [60, 61, 62]. Motor imagery refers to an imagined movement of certain parts of the body that is not actually physically executed. In the context of our BCI, we will use motor imagery to acquire a binary (0 or 1) signal from the users by asking them to imagine moving either their left or right hand. It is well known that this

type of motor imagery can be characterized by changes in EEG power within the Mu (8-12 Hz) band between the left and right sides of the brain. Figure 5.1 illustrates the spatially-distributed nature of these power-changes in more detail: the top image depicts a spatial heatmap of how motor imagery leads to spatial changes in Mu frequency power. When the subject imagines a left-side movement, the Mu band experiences an increase in spectral power on the left side of the brain. Similarly, upon right-side imagined movement, the Mu power on the right side of the brain increases. The bottom image is an example of an actual motor imagery task carried out in a laboratory setting, where the subject is attempting to use motor imagery to mentally move the ball on the screen either to the left or right. For this application, the exact details of how classification was performed on the raw EEG signals can be found in previous work from our group in [50].

5.1.2 Human Computer Interfaces and Optimizing Communication Efficiency

Here we will discuss the information model that we will be structuring our BCI on, and how it lends itself to maximizing the efficiency of our communication channel. We begin the discussion by first assuming a single-user BCI for simplicity. In the most general sense, a human-computer interface is a system that facilitates signaling between human and computer agents in some meaningful way. The computer can relay sensory information to the user about what it interpreted so that subsequent signaling can be corrective in nature. This causal feedback loop allows the human and computer to cooperate via an iterative process, as shown in Figure 5.2.

In our case, we will focus on the specific scenario where the human agent, modeled as an "encoder", is modeled as possessing some form of abstract information or knowledge, which can be represented as $W \in \mathsf{W}$, that they would like to reliably convey to the computer agent, modeled as a "decoder", over a noisy channel with feedback. This abstract piece of knowledge is sometimes referred to as the "message point". Also note that the message point

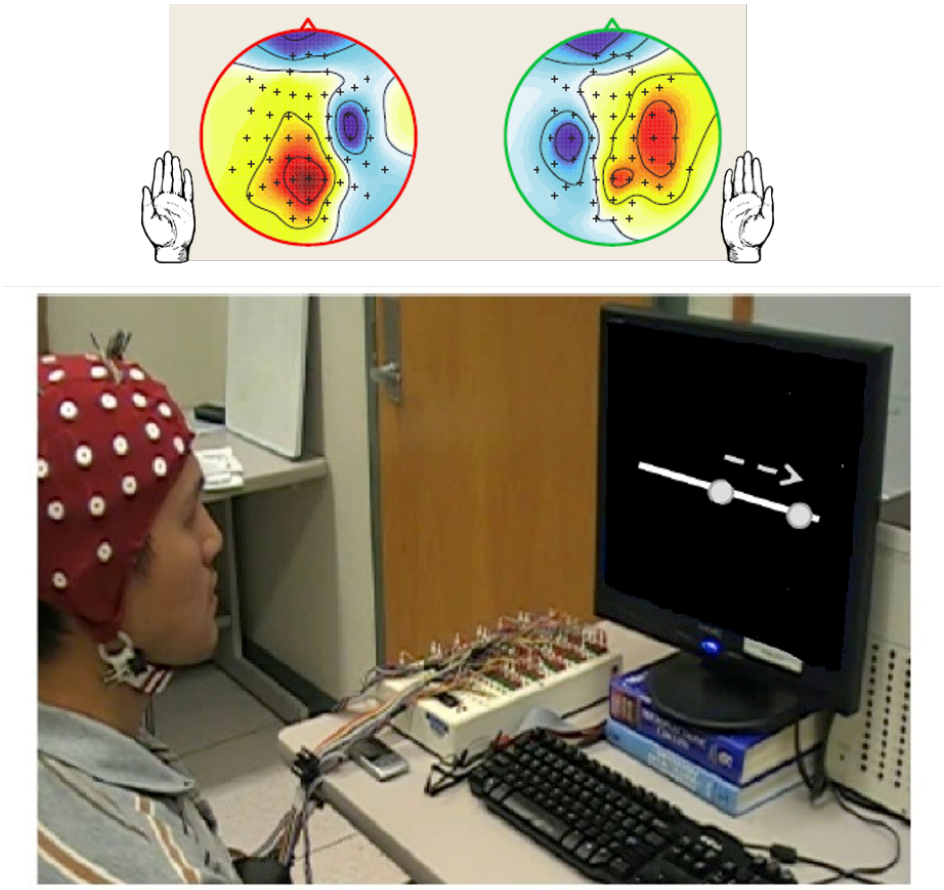


Figure 5.1. Top Image: A heatmap representing changes in spectral power across the EEG array as motor imagery is performed on either side. Bottom Image: A laboratory example of a real-time motor imagery task.

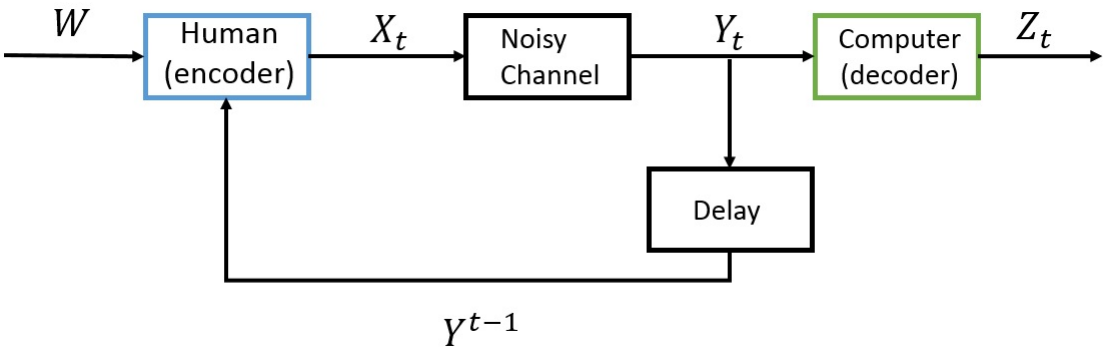


Figure 5.2. The basic encoder/decoder-based communication framework

does indeed lie in the space over which our optimal transportation map will operate, and thus it remains consistent with the notation from earlier in Section 2.1. Furthermore, at every time step t, the human agent iteratively generates input signals to the noisy channel, X_t, as a function of the underlying message point, and feedback of all the information the computer has causally seen so far, which we denote as Y^{t-1} (the superscript denotes "all Y up until time $t-1$"). Each output of the noisy channel is governed by a probabilistic model, $P(Y|X)$, and gives rise to the state variable, Z_t, updated by the computer agent. This state variable can manifest itself in a variety of ways depending on the application, e.g. Z_t could represent the actuation of a signal in the environment or on an external device, or it may represent a continuously updated probability distribution, etc. Thus, we can say that the encoder is governed by a collection of time-varying encoding functions $e = \{e_t : \mathsf{W} \times \mathsf{Y}^{t-1} \to \mathsf{X}\}_{t=1}^{T}$, and the decoder governed by a collection of time-varying decoding functions, $d = \{d_t : \mathsf{Y}^t \to \mathsf{Z}\}_{t=1}^{T}$, whereby each channel input X_t and decoder state Z_t is given by:

$$X_t = e_t(W, Y^{t-1}) \tag{5.1}$$

$$Z_t = d_t(Y^t) \tag{5.2}$$

5.1.3 Connection To Team Decision Theory

The human and computer are cooperating to achieve a common goal, and this can be manifested from the lens of team decision theory as a joint maximization of a reward function, subject to constraints on the information structure:

$$(e^*, d^*) = \arg\max_{e,d} \sum_{t=1}^{T} \mathbb{E}\left[r_t(W, X^t, Z^t)\right] \tag{5.3}$$

$$\text{s.t. } X_t = e_t(W, Y^{t-1}) \tag{5.4}$$

$$Z_t = d_t(Y^t) \tag{5.5}$$

These classes of decentralized control problems fall into the category of "non-classical information structure" problems that are in general notoriously "difficult" to solve [34]. However, for certain types of reward functions, information-theoretic convex relaxations can be tight and solve the original problem [44].

For certain subclasses of such team decision problems, it is known that there exist optimal signaling schemes for which the encoder and decoder strategies do not vary with time, and obey the following minimal structure [30, 77, 7]:

$$x_t = \bar{e}(w, z_{t-1}) \tag{5.6}$$

$$z_t = \bar{d}(z_{t-1}, y_t) \tag{5.7}$$

The important thing to note here is that the encoder strategy $\bar{e} : \mathsf{W} \times \mathsf{Z} \to \mathsf{X}$ still represents the strategy embodying the human agent, and similarly, the decoder strategy $\bar{d} : \mathsf{Z} \times \mathsf{Y} \to \mathsf{Z}$ still represents the strategy embodying the computer agent; the distinction now is that both $\bar{e}$ and $\bar{d}$ are time-invariant, yet performance of this communication framework is still optimal from an information-theoretic perspective.

5.1.4 Posterior Distribution as a State Variable

For this application, we will primarily focus on the class of reward functions pertaining to maximizing how much uncertainty about W is being reduced on average with every use of the communication channel. More specifically, we consider maximizing the mutual information between W and the observations Y^T, which is equivalent to minimizing the amount of uncertainty of W, given Y^T (e.g. minimizing posterior entropy) [75, 37]. As such, we now consider $\mathsf{Z} = \mathsf{P}(\mathsf{W})$ to be the space of probability distributions over W, and Z_t can be treated as the posterior distribution of W given Y^t. We can then define the sequential information gain reward function as [30]:

$$r_t(W, X^t, Z_t) = \log \frac{dZ_t}{dZ_{t-1}}(W) \tag{5.8}$$

Because of the sequential nature of Bayes' rule, if we let $\pi_t = Z_t$ be the posterior distribution over W at time t, we can define a decoder strategy consistent with the structure of (5.7):

$$\pi_t = \bar{d}(\pi_{t-1}, Y_t) \tag{5.9}$$

$$\bar{d}^*(\pi_{t-1}, Y_t)(dw) \triangleq \frac{\pi_{t-1}(dw)P(Y_t|\bar{e}(w, \pi_{t-1}))}{\beta} \tag{5.10}$$

where β is the standard Bayesian normalization constant. As such, for any encoder strategy in this Bayesian scenario, it follows that:

$$\sum_{t=1}^{T} \mathbb{E}\big[r_t(W, X^t, Z^t)\big] = \sum_{t=1}^{T} I(W; Y_t|Y^{t-1}) = I(W; Y^T) \tag{5.11}$$

In terms of maximizing the mutual information between W and Y^t for any time step t, it is probably the easiest approach to reason about how this can be done in the context of a specific noisy channel model. In our case, we first look at the well-known Binary Symmetric Channel (BSC) studied by Horstein [35]. The capacity of a noisy channel defined by the distribution $P(Y|X)$ is given by the maximum mutual information between X and Y over all possible input probability distributions $P(X)$ [19]:

$$C = \max_{P(X)} I(X; Y) \tag{5.12}$$

and we designate the maximizer of this quantity as $P^*(X)$. For many commonly-used channels, $P^*(X)$ is well known. For a BSC in particular, the mutual information maximizing input distribution is the uniform distribution over the input alphabet into the noisy channel, i.e. $P(0) = P(1) = 0.5$. We will discuss the BSC model in more detail as we go further.

The mutual information of our model is upper-bounded by the capacity:

$$\frac{1}{T} I(W;Y^T) \leq C \tag{5.13}$$

where equality holds if and only if [19, 70]:

- X_t is statistically independent of Y^{t-1}
- $X_t \sim P^*(X)$

Therefore, all we need to do is ensure that both of these properties holds, and we can guarantee that we are achieving communication capacity. Within this context, maximizing mutual information is a necessary condition to guarantee that the posterior π_{t-1} converges to a point mass at W as rapidly as possible; in many situations, given other technical conditions, it is also sufficient [70]. The original posterior matching scheme in [70], as well as our group's generalization of the scheme to arbitrary dimension [48] which this section is based on, attains the upper bound of (5.13) with equality.

5.1.5 Human Friendly Feedback

As we have modeled the human user as an encoder within this framework, our expectation at this point is that the human must present an encoder strategy of the form $X_t = e(W, Z_{t-1})$; in other words, the human must decide on the next noisy channel input based on knowledge of the message point in conjunction with knowledge of the decoder's posterior distribution on W. In practice, this is extremely inconvenient for the human user; how would a human being keep track of a posterior distribution, or otherwise know how to reason about it in real time in order to come up with an encoding strategy to determine X_t? This is quite a tall order indeed, and so we should decide on a way to make this process more tractable from a human perspective.

In Figure 5.3, we have made a few modifications to the encoder/decoder scheme. Rather than imagine the human as the entire encoder module, we partition the responsibilities of the

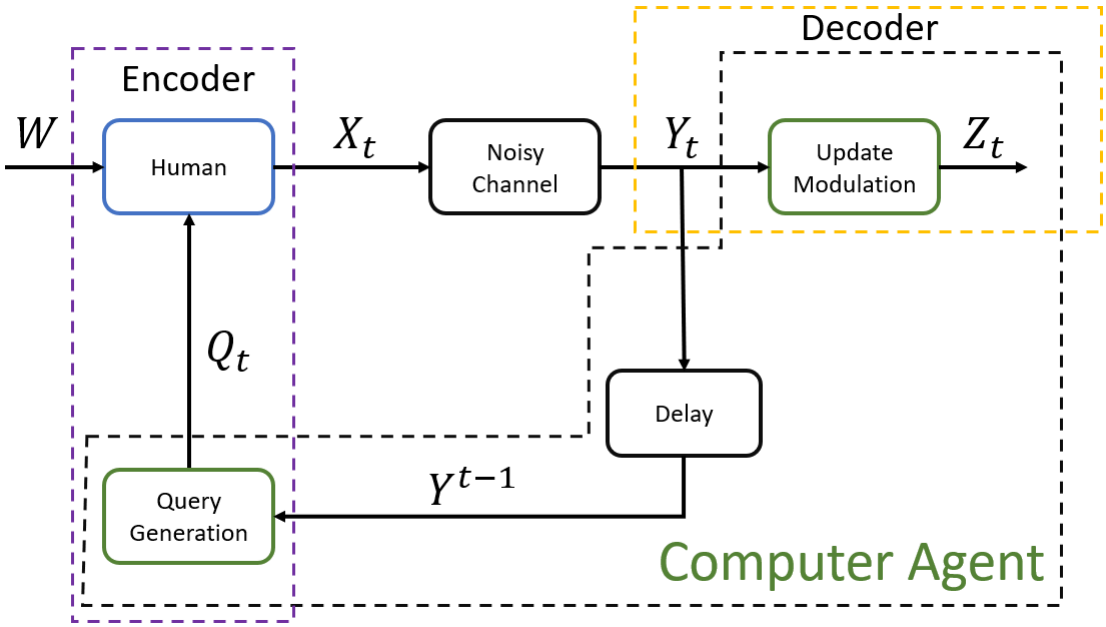


Figure 5.3. A simplified communication model based on the discussion in [73].

encoder between the human and computer agent (the blue solid box represents the human, and the 2 solid green boxes represent computer agent tasks). Here, instead of directly giving the human the entirety of the previous noisy channel outputs and/or the entire posterior Z_t, we choose to assign the computer agent the task of condensing the knowledge of Z_t to a simpler piece of information, $Q_t \in \mathsf{W}$, which we call the query point. The dotted purple box represents the true encoder module, given the traditional communication scheme, and the dotted orange box the traditional decoder; these two boxes have been added to the figure to emphasize the fact that the traditional communication model we have discussed up until this point is still a valid modeling framework for our system, and we have just re-allocated the more fine-grained responsibilities between the human and computer agent. In this model, the human may now just use the combination of W and Q_t to come up with their encoding strategy, which we can then design to be much simpler compared to forcing them to work with the entire posterior.

Continuing on with our BSC example, we will now make the above discussion more concrete by providing a specific application of the posterior matching scheme in the human-

computer interface setting. Consider a BSC model with crossover probability ε:

$$P(Y = y|X = x) = \begin{cases} 1-\varepsilon, & \text{if } y = x \\ \varepsilon, & \text{if } y \neq x \end{cases} \tag{5.14}$$

and $\mathsf{W} = [0,1]$, and $\mathsf{X} = \mathsf{Y} = \{0,1\}$. The human user in this situation would update their channel input X_t at every time step t given access to the message point, W, and the query point, Q_t. As X_t is, by definition, a function of W, the noisy channel model *also serves as the likelihood model* in our Bayesian update of the posterior π_t.

To ensure that $X_t \sim P^*(X) = [0.5, 0.5]^T$, we can simply define a feedback rule for the computer agent that dictates a query point that tries to guarantee an equal probability of X_t being a 0 or a 1. The query point is an embodiment of the information gathered thus far, and generated by the computer agent as a form of feedback for the human to respond to. In order to guarantee that $X_t \sim P^*(X)$, we can designate the following very simple query and response rule for the computer agent and human, respectively:

$$Q_t = \text{median}(\pi_{t-1}) \tag{5.15}$$

$$X_t = \begin{cases} 0, & W \leq Q_t \\ 1, & \text{otherwise} \end{cases} \tag{5.16}$$

In this case, at every time step the computer agent need only compute the posterior distribution and relay the median of the distribution to the human agent as Q_t. From the perspective of the human, their task becomes almost trivially simple: given Q_t, they simply signal whether the message point W is greater than or less than Q_t. This simple comparison yields the uniform input distribution we desire as well, as W has equal probability of lying on either side of Q_t from the perspective of the posterior. Perhaps the most important interpretation

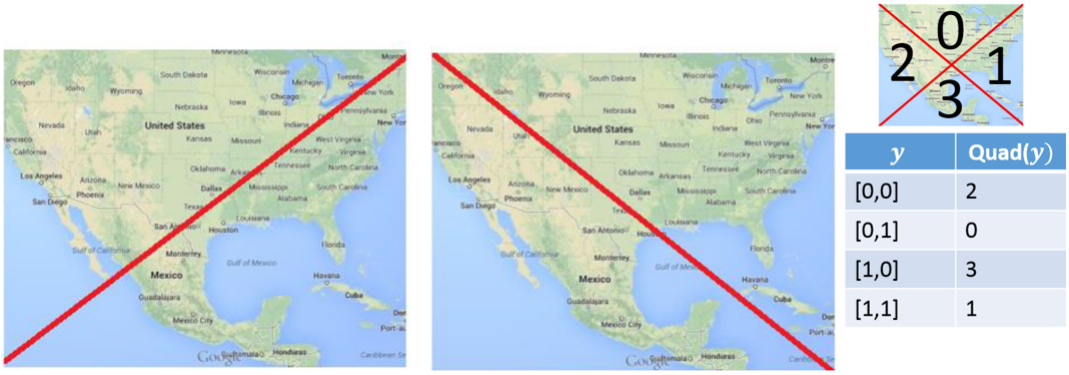
Figure 5.4. A 2-dimensional version of the BSC BCI described in Section 5.1.2. At every time step, the query point Q_t is updated based on user responses, and as such, so are the dividing lines. The large numbers simply represent indices for identifying each quadrant.

of Q_t is this: it is the *most informative question* the computer can ask the human to the end of learning what W actually is in the case of a BSC. Another more computer science-oriented way to potentially interpret this scheme is as a binary search on the [0,1] line *in the presence of noise*, and the probabilistic model must be integrated into the problem to compensate for the effect of the noise of the communication channel.

5.1.6 Posterior Matching in 2 Dimensions with Optimal Tranportation

In this section, we will discuss how to generalize the previous scheme to a 2-dimensional demonstration application: a multi-user BCI whose aim is to signal a message point in $\mathbb{R}^2$ shared by the users to a computer agent using only binary motor imagery-based EEG signals. Consider the user interface in Figure 5.4. Here, we have overlaid 2 intersecting lines dividing a map of the United States into 4 partitioned regions, and we define the query point Q_t as the intersection of these two lines. Therefore, the query point changes at every iteration, as do the dividing lines. Assuming the human users have a shared location of interest in mind, W, that they would like to signal to the computer, the goal of the users is to interact with the computer agent in some way in order to ultimately move the query point precisely to the location of interest using their binary EEG signals.

Given a query point Q_t in this context, we then denote the 4 quadrants that we have divided the query space into as $\Gamma_{k,t}$ for $k = 0 \ldots 3$, and denote the north quadrant as index 0,



y	Quad(y)
[0,0]	2
[0,1]	0
[1,0]	3
[1,1]	1

Figure 5.5. An example of how we can split the problem into two sub-problems, and hand 2 users 1 sub-problem each. Based on the combination of the responses from the two users, we can infer which quadrant in the overall problem W lies in based on the values in the table.

the east quadrant as index 1, the west quadrant as index 2, and the south quadrant as index 3.

This application is an extension of the BSC example in the sense that it can still be seen as a symmetric channel, but in higher dimension. With the partition notation above in mind, the goal of the users is then simply to signal which quadrant W lies in at every iteration, which is in a sense an analogous task to the "higher or lower than 0.5" signal from the BSC example, except in 2 dimensions. In practice, we can then split the task in half as depicted in Figure 5.5, one user receiving the image of the positive sloping line, and one receiving the image of the negative sloping line. The task for each user then simply becomes to designate whether the message point lies to the left (Y=0) or right (Y=1) of the dividing line they are shown. Given the response from both users, we can then infer which quadrant within the overall problem the message point lies in, based on the table in the figure.

In the case of arbitrary dimension symmetric channels, the mutual information maximizing input distribution into the noisy channel is still the uniform distribution, and as such, we must once again guarantee that the probability of all user responses into the noisy channel are equal in order to achieve capacity. In general, this can be difficult to do, as it is not immediately clear as to how to divide the query space into 4 quadrants of equal probability mass from the perspective of the posterior distribution π_{t-1}, which is now a probability measure in 2

dimensions as well. Indeed, we can use optimal transportation to assist us here.

Consider a scenario where we build an optimal transportation map S_t that pushes from π_{t-1} to the prior $P(W)$, which we assume to be uniform. In other words, from our previous discussion on optimal transport we will be assigning $U = \pi_{t-1}$, and $V = P(W) =$ uniform. One way we can interpret the action of this map is depicted in Figure 5.6. At the end of timestep t, we compute the posterior distribution π_{t-1} based on the observation of Y_{t-1}. The red dots in the figure designate the query point that would achieve a partition of the mapping into 4 quadrants of equal mass. Our map, by construction, "warps" space from the updated posterior space back to a uniform space. In this situation, it is clear which query point in this uniform, newly-warped space divides the space into 4 quadrants of equal probability mass: it is simply the center of the image.

Technically speaking, the users can then do the following in the next time step: the users can use S_t to warp the query point W to a new point W_t, and signal to the machine in which quadrant of the newly warped uniform space W_t lies in. This is depicted as Scenario 1 in Figure 5.7. However, the act of using S_t to push W to W_t would be extremely difficult for the human to do in practice, i.e. they would have to know how to warp their own message point in their head, or somehow otherwise match an image of the warped space with where the message point is, either of which is a highly impractical thing to do. So we use an alternative method: rather than have the user warp the *message point*, we instead ask the computer agent to warp the *query point*.

To accomplish this, we can use S_t^{-1} to transform the midpoint of the original unwarped space to the newly warped posterior space. This makes sense since the map was originally designed to push from the posterior to a uniform space; therefore, the inverse map will push from the uniform space to the posterior space, and by warping the midpoint of the image through the inverse map, we should get the corresponding point in the posterior space that divides the space into 4 quadrants of equal mass, which we can then use as our new query point.

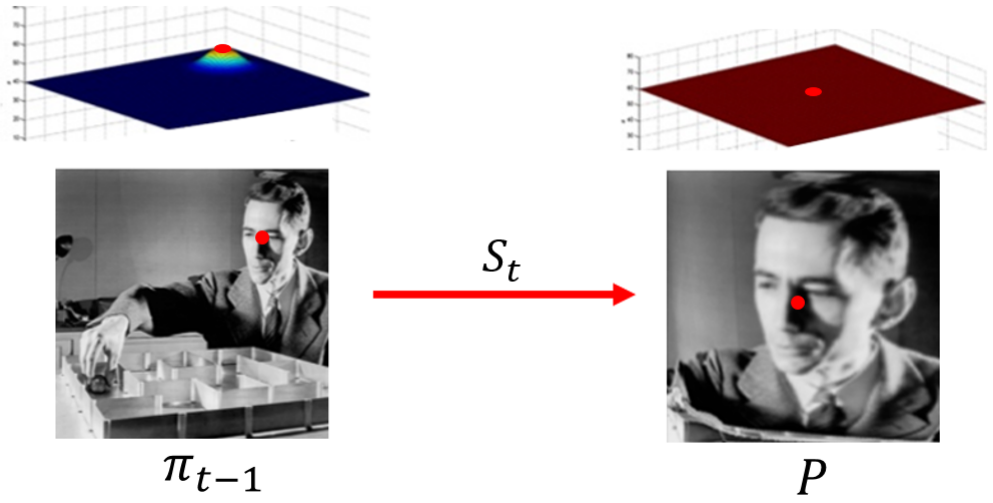


Figure 5.6. When we push from π_{t-1} to P, we effectively "stretch" out the mass around the peak in π_{t-1}, to make the space uniform; this "stretching" is illustrated by the stretching of the image of Shannon.

This mapping process, which we ultimately use in our live system as demonstrated in Figure 5.4, is depicted in Scenario 2 of Figure 5.7 and is by far the more convenient and intuitive process from the perspective of a human user.

Finally, after implementing the above framework in fully working real-time code, we carried out the experiment in conjunction with our collaborators at the University of Puerto Rico, Mayagüez. Figure 5.8 depicts what the C++ client program's user-interface looked like to our 2 subjects, one located in San Diego, one located in Puerto Rico. Before the experiment began, the subjects agreed on a message point on a map of the UCSD campus to specify to the computer agent. As described in the section above, each subject was shown only one of the dividing lines out of the two, and was asked to use their motor imagery identify whether or not the point of interest lay to the left, or right of the line they were shown. Figure 5.9 shows the interface as seen from a master server program coordinating the efforts between the two client programs. The red dot designates the agreed-upon message point, and the images to the right of each map shows the current state of the posterior distribution. After only 9 communication iterations, the posterior had essentially converged to a point mass, and the dividing lines had

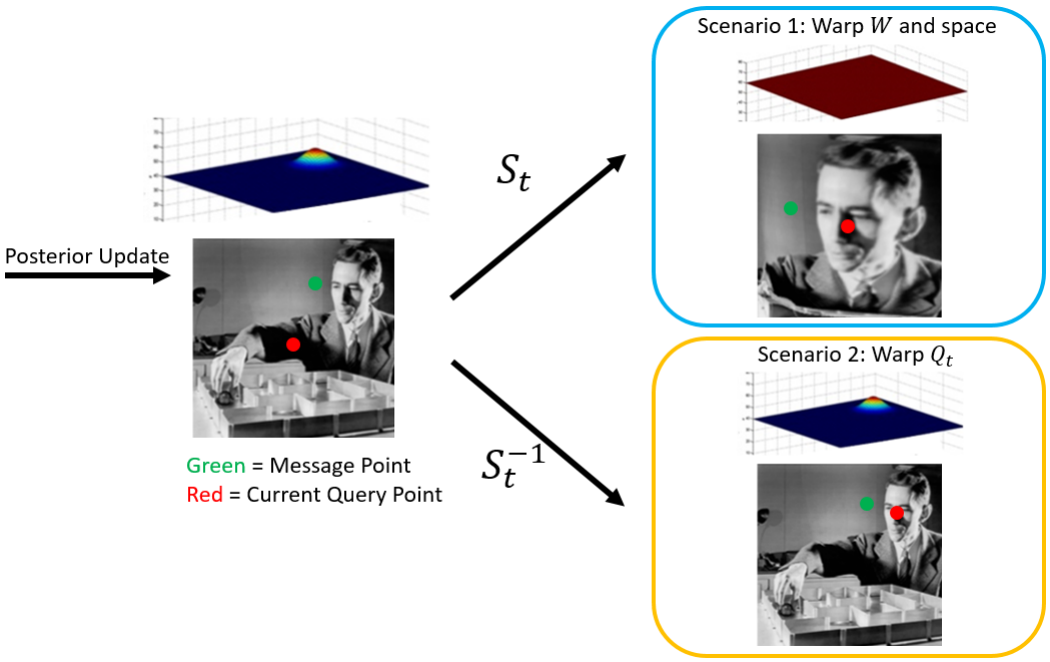


Figure 5.7. A depiction of the two, mathematically equivalent, scenarios of updating the query point. In Scenario 1, we warp the message point. In Scenario 2, we warp the query point instead, thus leaving both the space, as well as W, unwarped.

centered on the point of interest.

5.1.7 Acknowledgments

I would like to acknowledge Dr. Rui Ma, Dr. Sanggyun Kim, and Dr. Diego A. Mesa of the Coleman Lab for all of the contributions and helpful discussions pertaining to this project. As this was one of the earliest projects I developed as a member of the Coleman Lab, these fellow lab members were were extremely valuable sources of knowledge and insights into the problem.

I would like to acknowledge our collaborators/co-authors, Dr. Vidya Manian, Cristian Alzate, and Jaime Camacho, of the University of Puerto Rico, Mayagüez, without whom the brain-computer interface application described in [73] and in this section would not have been possible, both in terms of investigation into the EEG classifier, as well as offline analysis of preliminary motor imagery data we recorded with our labs' amplifier hardware.

Figure 5.8. Our 2 subjects connected to their associated client interfaces to perform the multi-user BCI task.

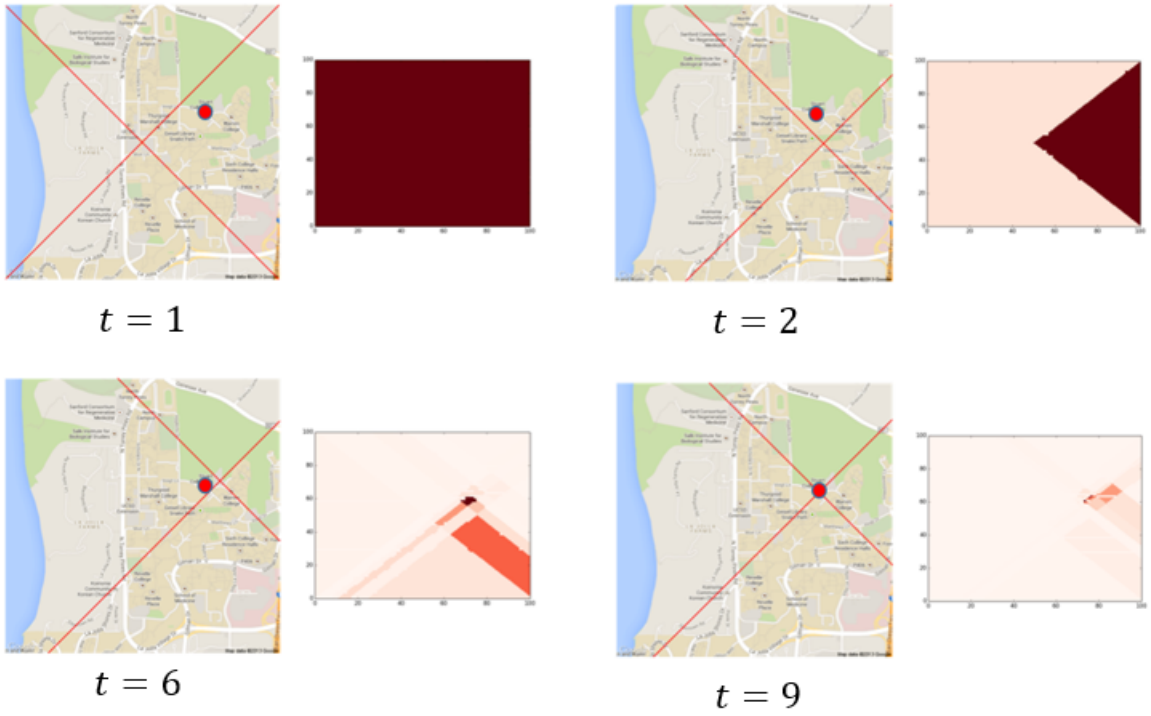


Figure 5.9. A depiction of the live experiment's progress over time. By communication iteration 9, the posterior had mostly already converged on the message point, showing the effects of the maximized communication rate.

5.2 Density Estimation of EEG Sleep Study Data

In this section, we will describe an application of our optimal transportation framework to density estimation in the context of automated sleep staging based on single-channel EEG recordings. This work was originally reported in 'Diffeomorphism learning via relative entropy constrained optimal transport", GlobalSIP 2016, Coleman, Todd P.; Tantiongloc, Justin; Allegra, Alexis; Mesa, Diego A.; Kang, Dae; Mendoza, Marcela, 2016, [16], in which the dissertation author was the primary investigator as well as presenter at the GlobalSIP conference in 2016.

If we've generated a transport map, S, one useful application of the mapping is to perform density estimation of a sample $x \sim U$, using Equation (2.2):

$$u(x) = v(S(x)) \det J_S(x)$$

In this situation, we assume that we have training samples drawn from U (e.g. having observed samples from some natural phenomena or experiment), and we would like to perform density estimation either on those samples, or samples we may acquire in the future. To accomplish this, we can then construct a mapping S that pushes from the unknown U from which our training samples are drawn, to some distribution V that we intentionally select to be easy to handle analytically, such as designating V as a standard Gaussian distribution, $V = \mathcal{N}(0, \Sigma)$. In this context, V becomes a "dummy distribution" that is purely used for the sake of computing the density estimate $u(x)$ using Equation (2.2) in conjunction with a properly constructed map $S\#U = V$. The natural advantage to using transport maps in this context is that there need not be any constraints on the structure of U beyond having continuous density, i.e. U can be highly multimodal, or otherwise arbitrarily shaped, and we can achieve good performance in our density estimation of U with a properly constructed transport map. Furthermore, we can take advantage of the scalability of our parallelized version of the map construction process to more comfortably scale the application in this context to potentially

high dimension.

As an example of the application of transport maps to density estimation, we will show results on an automated sleep-staging paradigm using single-channel EEG recordings, which are certainly indicative of our transport maps' ability to perform effective, multivariate density estimation. Identifying the different sleep stages a patient is in as they progress through a night of sleep can be beneficial for the treatment of many different conditions such as sleep apnea, insomnia, and narcolepsy [33]. Currently, however, sleep staging is performed manually by trained sleep technicians, and as such, involves a considerable amount of effort to score an entire night of sleep. Moreover, there is a non-trivial variability between different technicians' scoring methods, which hinders interpretability by clinicians [82]. In light of this, we will demonstrate here the ability to use transport maps to do binary classification on two pairs of sleep stages that are particularly interesting in terms of being able to differentiate one from the other: discerning the Wake vs. Light sleep stages (i.e. when the patient is more or less awake, vs. when they are in the beginning moments of sleep), and discerning the Light vs. REM (rapid-eye-movement, a slightly deeper stage of sleep relative to the Light stage) sleep stages. Historically, the Wake vs. Light classification problem typically nets high classification performance even using the most naive of models, as EEG signals associated with the Wake and Light stages tend to be very physiologically dissimilar from each other. However, the opposite can be said for discerning Light sleep stages from REM sleep stages, as they tend to look physiologically similar. Thus, we have chosen to evaluate our approach with one typically easy classification problem, as well as a relatively more challenging one. Moreover, this experiment will only use data recorded from a single, frontal, channel of EEG, as there is much motivation in a research context for increasingly non-invasive and non-intrusive in-home monitoring of sleep stages [39].

For this experiment, several patients have spent an entire night of sleep in a clinical monitoring setting. To turn this experiment into a supervised classification problem, we require

ground-truth labels for our recorded EEG data; to this end, a trained technician generated a clinical hypnogram for each patients' night of sleep, designating what they believed to be the patients' stage of sleep at each time point through the night. Using data collected from frontal electrodes placed on the forehead of each patient for the duration of sleep, we partitioned the data into 30-second time-windows, and the power spectra from the alpha, theta, sigma, and delta frequency bands were extracted from each time-window, consistent with the feature-extraction process described in [39]. Thus, we are left with feature vectors in $\mathbb{R}^4$, and labels indicating what stage of sleep each feature vector (and thus each time-window) corresponds to.

We can integrate our transport mapping into this framework by learning the densities pertaining to each specific sleep stage, and then using knowledge of these densities to perform a simple log-likelihood ratio classification test. We assume that the data gathered in our sleep study were sampled from some unknown densities conditioned upon the sleep state, U_{state}. We therefore need a transport map S_{state} to push from U_{state} to $V = \mathcal{N}(0,\Sigma)$. For two given classes of interest, we can then use Equation (2.2) to build a classifier using these transport mappings by performing a log-likelihood ratio (LLR) test with the LLR given as:

$$\begin{aligned}\text{LLR}(x) &= \log\left(\frac{U_A(x)}{U_B(x)}\right) \\ &= \frac{C_A}{C_B} - \frac{1}{2}(S_A(x)-\mu)^T\Sigma^{-1}(S_A(x)-\mu) \\ &+ \frac{1}{2}(S_B(x)-\mu)^T\Sigma^{-1}(S_B(x)-\mu) + \frac{D_A}{D_B}\end{aligned}$$

where A and B represent the class labels we are specifically interested in, and D_A and D_B represent $\log\det(J_{S_A}(x))$ and $\log\det(J_{S_B}(x))$, respectively. Thus, the two scenarios we will consider are:

1. A = Wake, B = Light

2. A = Light, B = REM

Using this framework, we performed 5-fold cross validation for each of the 2 binary classification problems, with a 20/80 test/training split over 4932 time-windows per class (total 9,864 data points) for the first problem, and 6592 time-windows per class (total 13,184 data points) for the second problem. Training sets were used in our optimal transportation framework to construct mappings S_{class}, $class \in \{\text{Wake}, \text{Light}, \text{REM}\}$, using fully-expressive polynomial parameterizations of order 5 (higher orders did not seem to significantly increase performance). We then compare the performance of an LLR classifier using our transport map-based density estimation to a very naive LLR classifier where densities are assumed to be Gaussian. For the densities that were assumed to be Gaussian, the training sets in each fold were also used to estimate the means and covariances.

Figures 5.10 and 5.11 show the performance of these two classifiers for both classification problems in the form of ROC curves for each of the 5 folds, as well as an average curve for each problem and classifier. Under the naive Gaussian assumption, the LLR test for Wake vs. Light achieves an average area-under-the-curve (AUC, perfect performance would be 1.0) of 0.733. Even using such a naively structured classification model, this somewhat decent performance is expected given that the classification of the Wake vs. Light sleep stages is typically considered an easier problem. But even so, the transport map-based classifier achieved an average AUC of 0.916, a significant improvement over the naive classifier.

Similarly, in the Light vs. REM case, the average AUC for the Gaussian classifier is 0.532, which is representative of a classifier model that is basically guessing labels (an AUC of 0.5 represents a completely naive classifier that just guesses labels at complete random). Again, this makes sense, as this classification problem is much more on the challenging side. By comparison, the transport map approach achieved an average AUC of 0.806, again showing significant improvement in this challenging scenario, and using a relatively simple classification method like LLR.

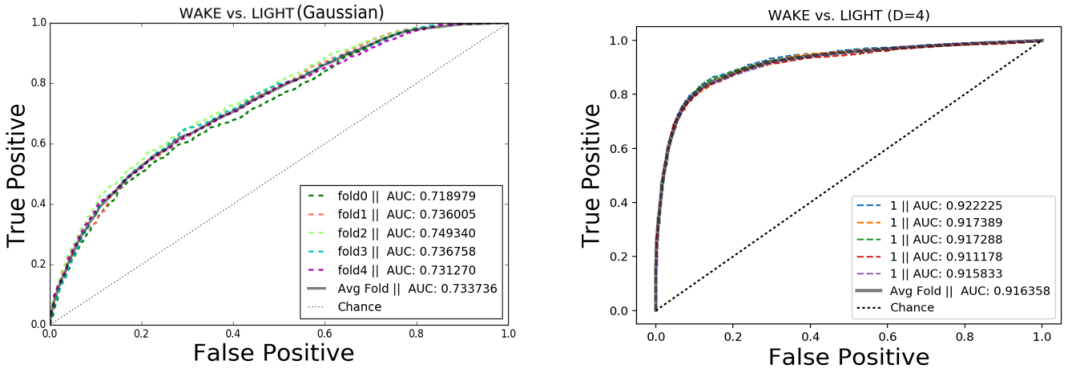


Figure 5.10. A comparison of performance of the naive Gaussian LLR classifier (left) and the transport map-based LLR classifier (right) on the Wake vs. Light problem.

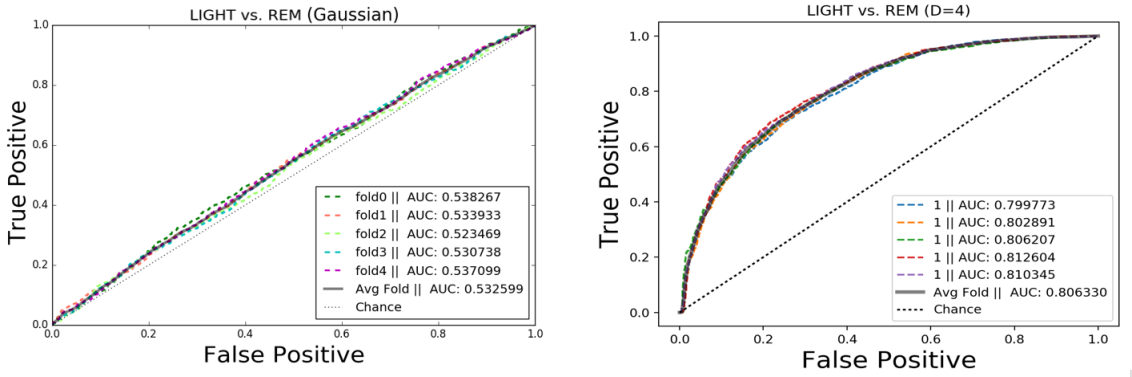


Figure 5.11. A comparison of performance of the naive Gaussian LLR classifier (left) and the transport map-based LLR classifier (right) on the Light vs. REM problem.

From these two proof-of-concept test cases, it is clear that the transport map approach does quite a reasonable job at estimating the class specific densities for this problem, and shows that it could potentially be viable for other application areas involving density estimation of arbitrarily-shaped densities as well.

5.2.1 Acknowledgments

I would like to acknowledge my co-authors of [16], Alexis Allegra, Dr. Diego A. Mesa, and Marcela Mendoza, for all their contributions to this work. I would also specifically like to acknowledge Dr. Dae Kang, as he originally performed all the collection of the EEG data for his own work [39] before we collaboratively recognized the potential application for our

optimal transport framework.

5.3 Generative Modeling of MNIST Data

In this section, we present a demonstration of our transport map framework's ability to generate maps for transporting relatively high-dimensional data with reasonable model accuracy with a generative model for the MNIST handwritten digits dataset [45]. This application was originally described in "A distributed framework for the construction of transport maps", Mesa, Diego A.; Tantiongloc, Justin; Mendoza, Marcela; Coleman, Todd P., which has been submitted to Neural Computation in 2018,[52], of which the dissertation author was a primary co-investigator.

Generative modeling problems entail observing a large dataset with samples from some unknown distribution U and attempting to learn a representation or model so that new independent samples from U can be generated. Perhaps some of the most popular recent methods of achieving this is through the use of deep neural networks, specifically variational autoencoders [43], generative adversarial networks [29], and auto-regressive neural networks. These models have consistently led to impressive results in a number of applications, but their tractibility and theory are still not fully developed. That being said, the purpose of this section is not necessarily to draw comparisons between our approach and previously developed methods for generative modeling. Rather, the purpose of this section is to present our transport map framework as a viable alternative that indeed has a rich, theoretical backbone to the formulation, and to motivate further study and comparison between optimal transportation methods and deep network-based methods for generative modeling (or perhaps some intersection of the two), both in terms of performance and scalability. Furthermore, this example demonstrates our ability to generate maps in relatively high dimension as far as optimal transportation is concerned, which further emphasizes how approaches like ours can scale with hardware availability.

Similar to the density estimation case, we assume that samples from each class of

MNIST data is drawn from some U_{digit}, where $digit$ denotes the MNIST written digit associated with the distribution. We then construct a sequential mapping, S_{digit} that pushes U_{digit} to a dummy distribution $V = \mathcal{N}(\mathbf{0}, \Sigma)$. In the density estimation case, we made the decision to designate V in this way as it allowed for convenient computation in Equation (2.2) to evaluate $u(x)$. However, in this scenario, the logic behind this decision is slightly different; we would like to use our mapping as a way to generate samples from U, and as such, if we have a distribution V that is easy to draw samples from, we can then push those samples through the **inverse** of our mapping to generate samples from U. More formally, we can draw samples $z_i \sim V$, and subsequently compute $S^{-1}(z_i) \sim U$. Because of this V can once again be any distribution of our choosing, but we choose a standard Gaussian distribution for the sake of being able to very easily draw samples from it.

Each image in the MNIST dataset is a 28x28 pixel image; therefore, by flattening each image into a vector of data, our maps will operate in $\mathbb{R}^{784}$. Each map we constructed comprised a 15-map sequence using the Single Variable Knothe-Rosenblatt parameterization, $\theta = 2$, and each sequential map being of order 2. To compute the inverse transformation, we used the methodology described in Section 4.3. Figure 5.12 shows the result of this process in the form of samples drawn from our mappings. Each row consists of 10 individual samples drawn from the corresponding U_{digit}, and most of the generated images seem to be properly representative points drawn from their respective distributions.

This example is particularly interesting as we can draw parallels between it and the demonstration of a similar generative model in [69]. Although the methodology is inherently very different in that work, with respect to both the objective functions used to solve the problem, as well as the parameterization using a deep network rather than a polynomial chaos expansion, it is extremely interesting to see optimal transport successfully carried out in such high dimension. As mentioned in Chapter 1, optimal transport is sometimes unsuccessful at being applied to problems of even higher than 100 dimensions, as many classical approaches

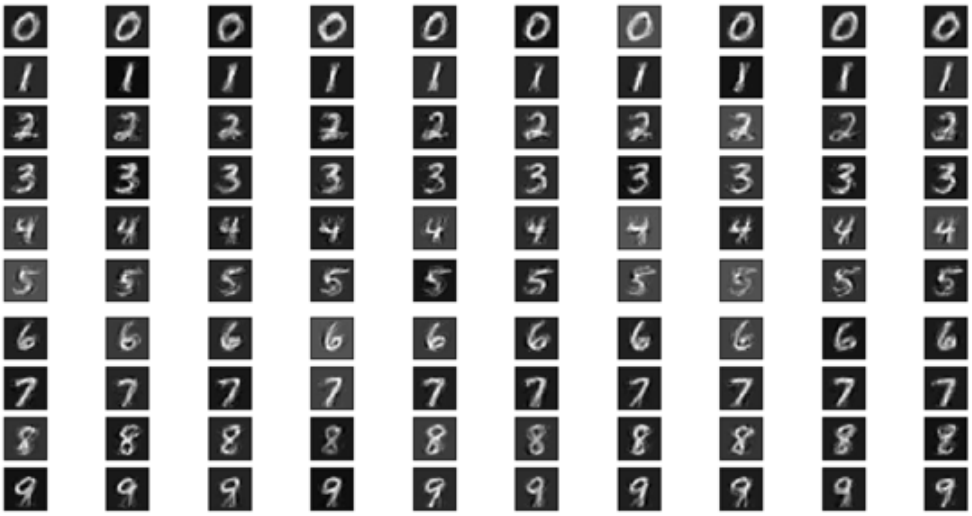


Figure 5.12. Samples drawn from each of the U_{digit} distributions using our inverse sequential maps

become quite computationally taxing as dimension increases, both in terms of time and space. But it is an exciting prospect to see how far parallelized methodologies that scale well with hardware such as our approach, and deep network-based parameterization methods such as Seguy et al's approach from [69] can take optimal transportation applications in the future.

5.3.1 Acknowledgments

I would like to acknowledge my co-first-authors of [52], Dr. Diego A. Mesa and Marcela Mendoza for all the helpful contributions and discussions pertaining to the GPU implementation that made this high dimensional problem possible. In addition, I would like to acknowledge Amazon Web Services for providing research funds that allowed us to take full advantage of their GPU compute instances, without which this problem would quite literally have been computationally impossible for us to solve.

5.4 Bayesian Inference for Preference-Based Deep Reinforcement Learning

In this section we will present preliminary results for an application of our optimal transportation framework to the field of deep reinforcement learning (DRL). We will discuss the current state of progress with respect to this application here, and then at the end of the section discuss potential future directions for the research to go. In addition, this application to DRL is actually also technically a Bayesian inference problem, and so this section serves a dual purpose of showcasing the Bayesian inference capabilities of the mapping process as well.

5.4.1 The Reinforcement Learning Problem

In reinforcement learning, the general goal is to teach a computer agent to behave within some environment in a maximally efficient way by allowing the agent to act within the environment and decide which actions are preferable in any particular situation [72]. This efficiency is normally quantified by the maximization of some notion of "reward" over time. Most modern reinforcement learning problems are formalized as Markov Decision Processes (MDPs) that involve the following key components:

- A state space, S, that denotes the set of all possible states the environment can be in
- An action space, A, that denotes the set of all possible actions the agent can take
- A policy, $\pi : \mathsf{S} \to \mathsf{A}$ that represents the agent's behavior in the environment, and maps states to actions (potentially stochastically, in which case, $\pi(a|s)$ is a probability distribution conditioned upon the current state)
- A reward function, $r : \mathsf{S} \times \mathsf{A} \to \mathbb{R}$, that represents a numerical reward given a state-action pair

- A model of the environment, $T(s'|s,a)$, where $s', s \in \mathsf{S}$, $a \in \mathsf{A}$, that represents the probability of arriving in state s', given an action a taken in a state s.

Within the environment, we discretize the agent's behavior into time steps; at every time step, t, the environment takes on a state s_t, and the agent takes some action a_t based on its policy π_t. The environment then changes its state based on the model T, the agent sees the new state, s_{t+1}, as well as receives a reward for performing its last action, $r(s_t, a_t)$, and takes another action based on the new state. This process continues either infinitely, or until some environment-specific termination condition is fulfilled. Given this definition of the problem, the overall goal of the agent is to find a policy that maximizes the expected reward over time over the distribution of potential initial states:

$$\arg\max_{\pi} \mathbb{E}_{\pi}\big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s\big] \tag{5.17}$$

where γ is a discount factor between 0 and 1. The maximizer of (5.17) will be π^*, the optimal policy with respect to the environment. Beyond the policy, there are two other functions of interest within the array of reinforcement learning computational strategies:

- The state-value function, $V_{\pi} : \mathsf{S} \to \mathbb{R}$. The V_{π} function represents the expected return of beginning in a state s, and then following the policy π from there for an infinite number of timesteps, or until termination. In other words $V_{\pi}(s) = \mathbb{E}_{\pi}\big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s\big]$ and can loosely be considered as how much value we assign to a state s overall with respect to the current policy.

- The state-action-value function $Q_{\pi} : \mathsf{S} \times \mathsf{A} \to \mathbb{R}$. The Q_{π} function is similar to V_{π}, except it is defined as the expected return associated with starting in a state s, taking an action a, and then following π starting the next timestep until infinity/termination. In other words $Q_{\pi}(s,a) = \mathbb{E}_{\pi}\big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a\big]$

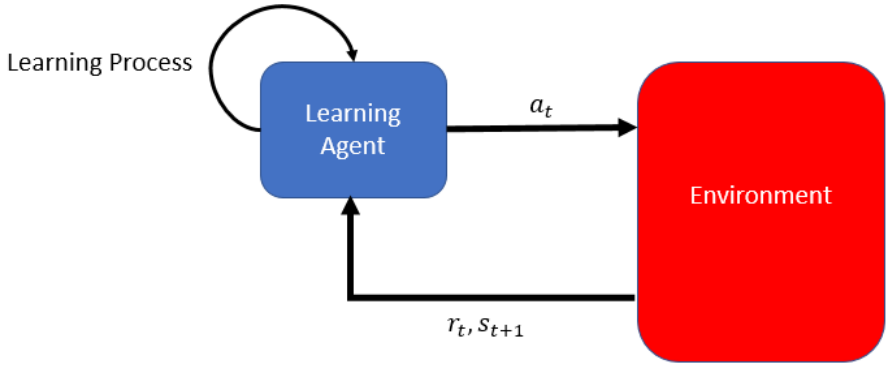


Figure 5.13. General reinforcement learning workflow. The learning agent takes an action in the environment given the state, the environment returns a reward and new state, and the agent engages in a learning procedure to refine its behavioral model.

There is a wide array of strategies designed over many years of research that aim to maximize Equation (5.17) but more often than not, they involve an iterative process where an agent continuously interacts with an environment, observes the rewards it receives for taking certain actions in certain states, and then iteratively refines an estimate of the best policy $\hat{\pi}$, or an estimate of the value function $\hat{V}$, or an estimate of the state-action value function, $\hat{Q}$, or some combination of those three quantities, to the end of finding a behavioral strategy that consistently nets it better and better rewards. The general reinforcement learning process is depicted in Figure 5.13. Furthermore, the recent resurgence in interest in deep networks has given rise to a relatively new sub-field within reinforcement learning known as Deep Reinforcement Learning, where deep networks are used as parameterized models representing some combination of $\hat{\pi}$, $\hat{V}$, and $\hat{Q}$, and occasionally, other components of the reinforcement learning framework as well [54, 55, 47, 53].

5.4.2 Preference-based Reinforcement Learning

The type of reinforcement learning problem we will be addressing is known as preference based reinforcement learning, and in this section, we will be focusing on the specific subset of preference-based reinforcement learning frameworks pertaining to learning an ap-

proximation of the reward function r. A more complete survey of all the other sub-areas of preference-based reinforcement learning can be found in [79].

Throughout the history of reinforcement learning research, many simulated environments have been developed as test-benches for the ever-evolving toolbox of reinforcement learning strategies, such as the widely-used OpenAI Gym toolkit [14]. For most of these simulated environments, a reward function is manually designed and implemented to accommodate learning the task at hand, and custom tailored to represent the desired behavior for the agent to learn when it is placed within a particular environment. However, there are a few potentially harmful issues with this approach:

- Reward Hacking [3] : When reward functions are handcrafted, sometimes agents can learn strategies that happen to maximize the reward function, without learning to qualitatively perform the task at hand. The classic example of this presented in [3] is where a robot is designed to learn to clean a room, and the reward function is designed such that the robot gains high reward when it sees less messes in the room. An example of a potential "hack" to this reward would simply be to cover all the messes in the room with some other object without actually cleaning them, which technically fulfills the reward criterion, but fails to accomplish the actual task.

- Reward Shaping [56] : When designing reward functions, especially in simulated environments, system designers and engineers may inadvertently encode desired behaviors into the reward function itself, thus steering the agent to the desired behavior in a way that is unfair relative to what the environment may be like in the real world. In [56], this phenomenon is termed "reward shaping", and can potentially lead to agents that perform well in simulation during training, but then fail to perform in the real world, as the simulated/training reward function was far too artificially informative.

Besides these two issues, there also exists the issue of reward functions potentially being far too complicated to design by hand at all, especially by engineers or machine learning scientists

who may not necessarily be experts in the task themselves. In this sitaution, we may rely on the knowledge of someone who *is* an expert at the task, and try to integrate their knowledge into the learning problem.

However, this approach presents its own set of complications: Imagine a scenario where a world-renowned tennis player was asked to handcraft a reward function for a tennis-playing environment so a robot could learn to play tennis. One problem could be that the tennis player may not necessarily be able to quantify their expertise in terms of states, actions, and rewards; i.e. sometimes when we develop an exceedingly high level of expertise in some task, we just "know" how to do the task well, and may not necessarily be able to put numbers to it. Secondly, the tennis player may be an expert in tennis, but they may not be an expert in reinforcement learning, and they may find it difficult to design a reward function that meshes well with the MDP-based framework in a technical sense. Machine learning scientists may often be successful in crafting reward functions for certain tasks like learning to balance a pole on a cart, or teaching a stick figure in a physics engine to walk or run (all of which are standard OpenAI Gym environments), simply because not only do they reasonably know how to do these tasks themselves, but they are also familiar with the intricacies of the reinforcement learning framework itself, and can more easily "meld" those two worlds together. For something like tennis, a tennis expert may know how to play the sport, but may not understand the reinforcement learning framework, and the machine learning scientist may understand the reinforcement learning framework, but not necessarily know how to play the sport, which leads to a mismatch of expertise in designing a training environment or computer agent.

Preference-based reinforcement learning is an attempt to mitigate the effects of all of these problems by allowing reward function approximations to be educated by human experts in a task environment, in as intuitive of a way as possible. The idea is to present the expert with some task that is minimally technical, yet is capable of allowing us to efficiently extract their expertise in some form, and integrate their knowledge into the design of our reinforcement

learning systems. To that end, our specific case will involve learning an approximation of the reward function which we denote $\hat{r} : \mathsf{S} \times \mathsf{A} \rightarrow \mathbb{R}$ from some kind of input from a human user or users (more precisely, [79] refers to this approximation as a "utility" function, as it is not fixed, and therefore induces a changing optimal policy over time, but we will not be making that specific distinction here, and just refer to $\hat{r}$ as the approximated reward function for simplicity). The most common modeling methodology for this approximation uses a linear sum of parameterized basis terms:

$$\hat{r}(s,a) = \sum_{k=0}^{K} \theta^T \phi_k(s,a) \tag{5.18}$$

where ϕ_k can be any desired sequence of basis terms or features of state-action pairs, and θ is a vector of parameters. In this scenario, we can imagine our learning task as attempting to infer the values of θ given some sort of interaction with our human user(s), which we define next.

In our learning framework, the learning agent will present a pair of simulated state-action trajectories, $\tau_1 = [s_1^1, a_1^1, s_2^1, a_2^1, \ldots s_T^1, a_T^1]$ and $\tau_2 = [s_1^2, a_1^2, s_2^2, a_2^2, \ldots, s_T^2, a_T^2]$, where T is some maximum simulation length. These trajectories can be simulated by the learning agent given its current policy estimate, $\hat{\pi}$, in the simulation environment. For each trajectory pair, (τ_1, τ_2), the human expert can then communicate which of the trajectories is preferable to the other, in terms of accomplishing the desired task at hand. Therefore, the human response can be interpreted as a binary response variable $Y \in \{1,2\}$, where $Y = 1$ implies that τ_1 is preferred over τ_2, and $Y = 2$ implies the opposite. Described more intuitively, the computer agent in this framework will show examples of what it thinks the desired behavior is, and the human expert will correct it, or reinforce it, based on those demonstrations.

Given M-many responses to M-many trajectory pair queries, we can them formalize the problem as computing the posterior distribution over θ given the responses: $P(\theta|Y^M)$. Indeed, this Bayesian inference strategy has been used in the past in this exact context, but

using a variety of loss functions, and/or likelihood functions to approximate the posterior [1, 80, 24, 79].

For our purposes, we will be using a sigmoid-based likelihood function inspired by [24] and [15]. The former uses the sigmoid function as a Bayesian likelihood as we will be, but the latter uses it as a loss term for a deep-network-based approximation of the reward function. In our scenario, we can interpret the likelihood as the following:

$$P(Y=1|\theta) = \frac{\exp(\sum_{t=0}^{T} \hat{r}_\theta(s_t^1, a_t^1))}{\exp(\sum_{t=0}^{T} \hat{r}_\theta(s_t^1, a_t^1)) + \exp(\sum_{t=0}^{T} \hat{r}_\theta(s_t^2, a_t^2))}$$
$$P(Y=2|\theta) = 1 - P(Y=1|\theta) \tag{5.19}$$

If we define $\Phi(\tau_i) = \sum_{t=0}^{T} \sum_{k=0}^{K} \phi_k(s_t^i, a_t^i)$ for $i = 1, 2$, then we can plug (5.18) into (5.19), to make the following simplification to the likelihood expression:

$$P(Y=1|\theta) = \frac{1}{1 + \exp(-\theta^T(\Phi(\tau_1) - \Phi(\tau_2)))}$$
$$P(Y=2|\theta) = 1 - P(Y=1|\theta) \tag{5.20}$$

With this formulation, if we interpret $\Phi(\tau_1) - \Phi(\tau_2)$ as a feature vector that is a function of the trajectory pair (τ_1, τ_2), we can interpret the entire problem as a whole as a Bayesian binary logistic regression problem over those feature vectors, with the trajectory preferences serving as class labels.

5.4.3 Transport Maps for Bayesian Inference

Given that this problem can effectively be reduced to an instance of Bayesian inference with a sigmoid likelihood model, we are finally ready to integrate our optimal transport map

framework. In a Bayesian inference setting, the posterior distribution is given by Bayes' Rule:

$$f(X|Y) = \frac{f(Y|X)f(X)}{\beta_Y} \tag{5.21}$$

where $f(X)$ is the prior, $f(Y|X)$ is a likelihood distribution, and β_Y is a normalization constant. In general, β_Y is very difficult to compute, which usually makes it difficult to draw independent samples from the posterior for the purpose of approximating it, computing expectations of it, or otherwise characterizing it. Historically, typical approaches to circumvent this difficulty and acquire (approximately) independent posterior samples involve Monte Carlo methods, such as Markov Chain Monte Carlo (MCMC) simulation [4], and indeed, previous works like [80, 24] do use these methods to approximate posterior expectations. However, if truly independent samples are desired, these methods tend to be computationally inefficient, as very long simulations need to be performed due to statistical dependency between sequentially simulated samples in the chain. Transport maps have previously been used to help mitigate this problem in the following broad sense [25, 49, 41]: if we construct a transport map S to push U to V, where we define U as a Bayesian prior, and V as the Bayesian posterior, we can draw independent samples from the prior (which should typically be easy to draw samples from), push it through S, and obtain statistically independent samples from the posterior. Assuming the computational process to identify S can be done quickly and efficiently, this process is often *dramatically* faster than using traditional sampling methods to draw from the posterior.

In the context of our optimal transport formulation, this involves a very simple extension to the objective function. Defining U as the prior, $f(X)$, and V as the Bayesian posterior, $f(X|Y)$, we combine Equations (2.2) and (5.21) and write:

$$f(X) = f(S(X)|Y)\det(J_S(X)) \tag{5.22}$$

$$= \frac{f(Y|S(X))f(S(X))}{\beta_Y}\det(J_S(X)) \tag{5.23}$$

For the purpose of optimization, we can safely ignore the normalization constant β_Y and directly plug the numerator from (5.23) into (2.8) to give us a new objective function:

$$\min_{\tilde{S}\in\mathfrak{D}_+} -\mathbb{E}_U[\log \tilde{u}(X;\tilde{S})] \tag{5.24}$$

$$= \min_{\tilde{S}\in\mathfrak{D}_+} -\mathbb{E}_{f(X)}[\log f(Y|\tilde{S}(X)) + \log f(\tilde{S}(X)) + \log\det(J_{\tilde{S}}(X))] \tag{5.25}$$

$$\approx \min_{\tilde{S}\in\mathfrak{D}_+} -\frac{1}{N}\sum_{i=0}^{N} \log f(Y|\tilde{S}(X_i)) + \log f(\tilde{S}(X_i)) + \log\det(J_{\tilde{S}}(X_i)),\ X_i \sim f(X) \tag{5.26}$$

Similar to our logic from Section 2.1, this problem also becomes a convex optimization problem as long as the prior and likelihood densities are log-concave. And finally, in the context of ADMM this special case only requires a very straightforward change of applying Bayes' Rule to the update of p_i in Equation (3.31):

$$p_i^* = \arg\min_{p_i} -\log f(Y|p_i) - \log f(p_i) + \frac{\rho}{2}||B^{k+1}\Phi_i - p_i||_2^2 + \gamma_i^{kT}(p_i - B^{k+1}\Phi_i) \tag{5.27}$$

Beyond this, the remainder of variable updates in our ADMM formulation, either for the fully-expressive mapping case or Knothe-Rosenblatt case, remain exactly the same.

5.4.4 Applying Transport Maps to Preference-Based Reinforcement Learning

In our reinforcement learning problem, the posterior is over the space of model parameters, θ, and we can construct a transport map from $P(\theta)$ to $P(\theta|Y^M)$ given the procedure described above by only modifying our p_i update. If we assume a uniform prior, we can ignore the $-\log f(p_i)$ term in (5.27). Furthermore, given that we have M-many trajectory pairs, $\eta_m = (\tau_1, \tau_2)_m$, $m = 1,\ldots,M$, accompanied by M-many user responses denoting trajectory

preferences, Y_m, $m = 1, \ldots, M$, we can plug in our sigmoid likelihood for $-\log f(Y|p_i)$ as:

$$p_i^* = \arg\min_{p_i} \; -\sum_{m=0}^{M} \left[\log f(Y_m|p_i, \eta_m)\right] + \frac{\rho}{2}||B^{k+1}\Phi_i - p_i||_2^2 + \gamma_i^{kT}(p_i - B^{k+1}\Phi_i) \qquad (5.28)$$

where the likelihood can be represented as a log-sum of the individual likelihoods over all our trajectory-pair queries, and the likelihood is expressed as being a function of the trajectory pair itself. One implementation detail we should mention is that since a sigmoid likelihood means we cannot derive a closed-form update, we must use gradient methods to optimize (5.28), which while straightforward to implement, does add to computation time. However, in practice, the algorithm is still relatively fast to converge as sigmoid functions are typically easy to optimize in this way anyway.

When all is said and done, the overall algorithm can be summarized in the following way, after random initialization of π_k and $\hat{r}$:

1. Use π_k to generate M trajectory pairs in given environment η_m, $m = 1, \ldots, M$
2. Have human expert rate trajectories and give responses Y_m, $m = 1, \ldots, M$
3. Use optimal transport to approximate map between prior and posterior with η_m and Y_m, for $m = 1, \ldots, M$, and use map to draw N samples from posterior and compute expectation of posterior as the new parameterization of reward function $\hat{\theta}$
4. Train agent's policy to $\hat{\pi}_{k+1}$ (or Q-function or V-function, depending on reinforcement learning algorithm used) using reinforcement learning algorithm of choice for T iterations with newly updated reward approximation $\hat{r}_{\hat{\theta}}$
5. Repeat from Step 1

Note in Step 4 above, the exact nature of the learning process depends on the learning algorithm used - because our transport map process isolates changes to only the reward

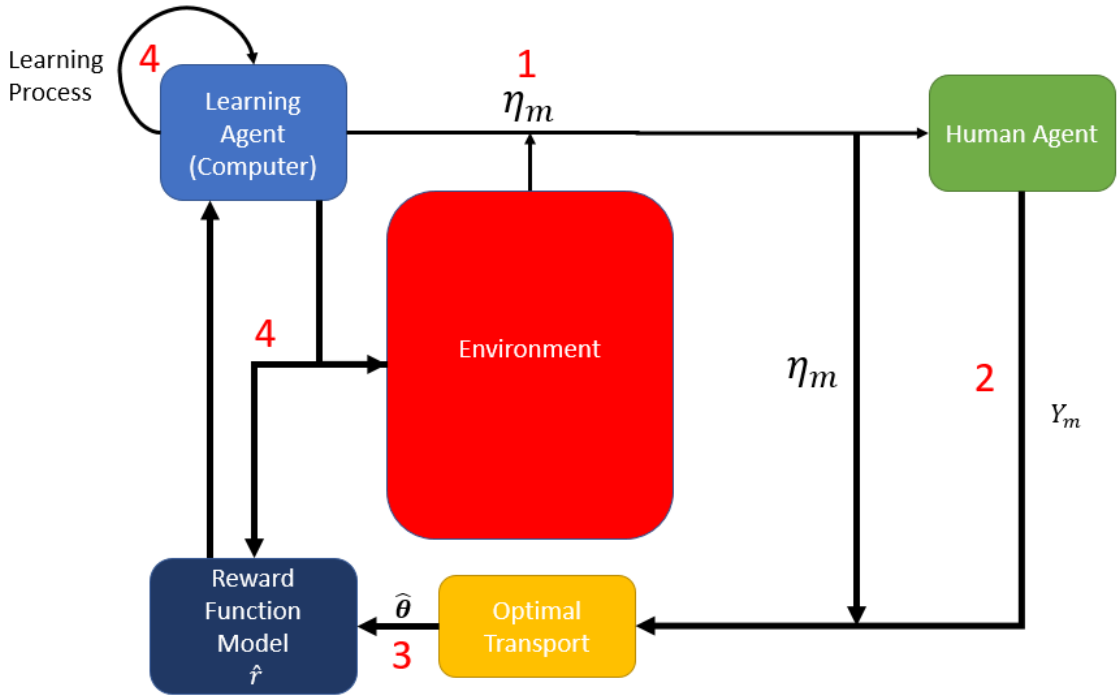


Figure 5.14. The preference-based reinforcement learning optimal transport workflow

function, this step can be populated in a plug-and-play fashion by any reinforcement learning agent of choice (e.g. Trust-Region Policy Optimization, Proximal Policy Optimization, Q-Learning, etc.) and the entire workflow should more or less be compatible. Thus, we have an alternating optimization problem where we alternate between partially optimizing $\hat{r}$, and then partially optimizing the computer agent using the reinforcement learning algorithm of choice. This workflow is graphically represented in Figure 5.14, with the red numbers indicating the corresponding step above.

5.4.5 Preliminary Results

In this section, we will discuss some of the preliminary results we achieved using the strategy described above. For this evaluation, we used the OpenAI Gym toolkit's InvertedPendulum-v2, InvertedDoublePendulum-v2, and Hopper-v2 environments based on the Mujoco library [74] as test cases (Figure 5.15). Furthermore, we leveraged the Proximal Policy Optimization (PPO) reinforcement learning agent strategy [68] to learn the policy and

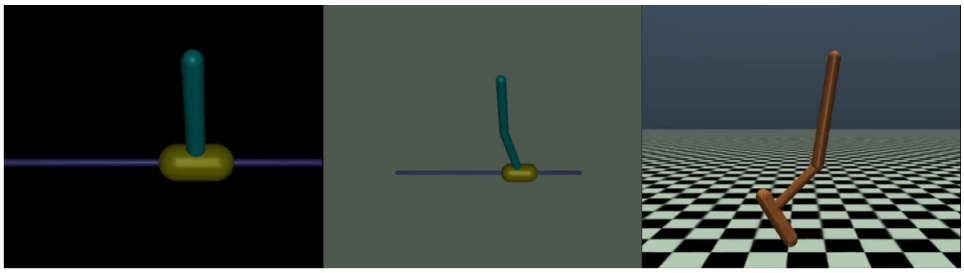


Figure 5.15. We tested our approach on these 3 environments of the OpenAI Gym toolkit - InvertedPendulum-v2, InvertedDoublePendulum-v2, and Hopper-v2.

value function for any given reward function approximation, as this is one of the most recent approaches that has achieved state-of-the-art performance on the OpenAI Gym toolkit as a whole. For all results, sample trajectories were rated by an oracle module that based its ratings on the true reward function encoded into OpenAI Gym. However, the oracle module also had a 10 percent chance of accidentally inverting its rating for any given trajectory pair, to simulate a notion of error that may come into effect when a human rater is used to designate preferred trajectories.

We began with the OpenAI-baselines [23] implementation of PPO, and made small modifications to make the agent fit more cleanly with our optimal transportation framework. However, we maintained the same hyperparameter settings described in [68] for use with Mujoco environments so that we could compare our performance to those in the publication. The reward function is treated as a function of both state and action, but the dimensionality of the mapping problem is also dependent on the choice of basis of the reward function approximation. For all the experiments discussed here, the reward function was parameterized as a fifth order polynomial with the same basis terms as the Single Variable Knothe-Rosenblatt multi-indices (i.e. no mixed multivariate terms in the basis). As a result, the dimensionality of the environments' state and action spaces, and the resulting dimension of our transport maps which we denote as D, given this reward function approximation are summarized as follows:

- InvertedPendulum-v2 : state dimension = 4, action dimension = 1, $D = 26$

- InvertedDoublePendulum-v2: state dimension = 11, action dimension = 1, $D = 61$
- Hopper-v2 : state dimension = 11, action dimension = 3, $D = 71$

For all maps, a sequence of 3 Single Variable Knothe-Rosenblatt models were used, $\theta = 1$. Figure 5.16 shows the performance achieved using our optimal transport-based framework, represented as a learning curve taking the average reward gained over the previous 100 episodes plotted vs the number of timesteps for learning agent training (a very common metric throughout the literature). Furthermore, each learning curve is itself an average over 5 runs of the algorithm in the given environment.

For InvertedPendulum-v2, the performance is comparable to most of the reinforcement learning agents in the literature. In this task, the maximum reward the agent can earn for any episode is 1000, which our agent achieves quite readily around 100,000 timesteps.

For InvertedDoublePendulum-v2, we actually achieve slightly better results than that of the vanilla PPO agent in [68]. Whereas the vanilla agent achieves roughly an average reward of around 7000 according to the publication, our agent extends towards an average reward of 8000.

However, for Hopper-v2, our performance falls short of the published PPO findings. Whereas the vanilla agent in [68] achieves roughly around 2000 of average reward after 500,000 timesteps or so, our achieves no higher than an average of 1500. While we are currently not entirely sure why this is not so, one hypothesis for why this is happening is pertinent to the explore vs. exploit dilemma [72]. The explore vs. exploit dilemma is a term describing a the tradeoff between an agent choosing to maximize rewards of the state-action combinations it has seen thus far in its experience within the environment, and an agent continuing to investigate state-action combinations that have not yet been seen in order to potentially uncover state-action combinations that yield even higher rewards.

A potential problem with our model is that our reward function approximation may require a fair bit more state-action space exploration than the vanilla agent that uses the true

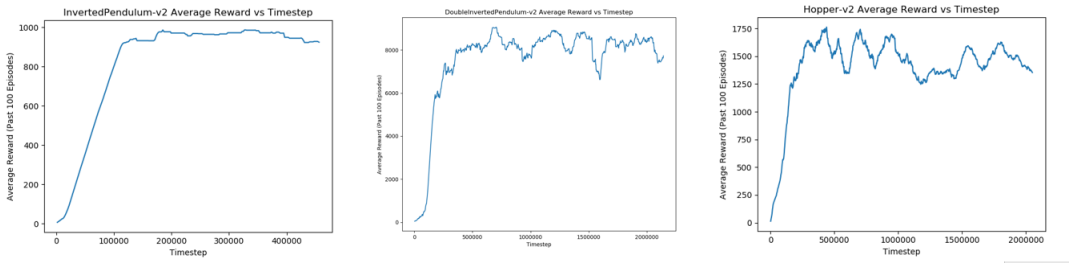


Figure 5.16. We tested our approach on these 3 environments of the OpenAI Gym toolkit - InvertedPendulum-v2, InvertedDoublePendulum-v2, and Hopper-v2.

reward function to learn the task, and our agent does not explore the state-action space nearly enough given its current implementation. In other words, in our case, the agent may need to explore the space so as to demonstrate trajectories to the human expert that are more informative than those it has demonstrated thus far; if the agent never even tries to demonstrate trajectories of higher value to the expert, the expert has no way of communicating that those trajectories are "better" than what it's currently already doing. Thus, our model may be more sensitive to the "explore" side of the explore vs. exploit dilemma than when using a perfectly informative, handcrafted reward function, which absolutely seems like a problem that can be remedied with the proper approach. One possible remedy to this problem may be adding a decaying noise model that adds a bit of perturbation to actions taken, and then levels off over time, so as to encourage further exploration of the state-action space to find state-action combinations that yield better results at least in the early stages of learning. Another possible solution is to experiment with different values of the clipping hyperparameter, ε, of the PPO algorithm, as different values can influence the desire for the agent to explore as well.

These results certainly seem promising to the end of applying optimal transport to reinforcement learning in this context. There is still much work to be done to truly realize the power of how transport maps might be useful here, but there are indeed directions of potential future work that may be of interest. More discussion on these issues can be found in Chapter 6.

5.5 Acknowledgements

This chapter contains applications originally described in “An information and control framework for optimizing user-compliant human computer interfaces”, Proceedings of the IEEE, Tantiongloc, Justin; Mesa, Diego; Ma, Rui; Kim, Sanggyun; Alzate, Cristian H; Camacho, Jaime J; Manian, Vidya; Coleman, Todd P., 2017, [73], of which the dissertation author was the primary investigator, “Diffeomorphism learning via relative entropy constrained optimal transport”, GlobalSIP 2016, Coleman, Todd P.; Tantiongloc, Justin; Allegra, Alexis; Mesa, Diego A.; Kang, Dae; Mendoza, Marcela, 2016, [16], of which the dissertation author was the primary investigator as well as presenter at the GlobalSIP conference in 2016, and “A distributed framework for the construction of transport maps”, Mesa, Diego A.; Tantiongloc, Justin; Mendoza, Marcela; Coleman, Todd P., which has been submitted to Neural Computation in 2018, [52], of which the dissertation author was a primary co-investigator.

Chapter 6

Conclusions and Future Work

In this work, we have presented an evolving line of optimization problems aimed at computing optimal transport mappings. Chapter 2 laid the foundation of a previously described Kullback-Leibler Divergence approach to forming the objective function for our optimization problem, Chapter 3 revisited the first parallelized formulation of the algorithm via ADMM, and then presented the novel version of the problem that takes advantage of the Knothe-Rosenblatt mapping and its properties, and further augmented that problem with a novel Wasserstein regularization term leading to a theoretically-backed means of performing sequential optimal transportation. Chapter 4 discussed some of the strategies used for the implementation of the framework, and details to give a more practical perspective on the algorithm. Finally, Chapter 5 showcased applications of this framework in a variety of contexts. First, we discussed the design of a multi-user brain-computer interface based on posterior matching. Next, we applied the framework to a density estimation problem for automated sleep-staging. Then, we demonstrated the framework in the context of a relatively high-dimensional generative modeling problem using the MNIST dataset. Finally, we concluded the chapter by showcasing some promising results for an application in preference-based reinforcement learning.

The field of optimal transport has a rich history of algorithms and strategies to acquire these elusive transport maps, all of which have their advantages and disadvantages. We will conclude this dissertation with a discussion on areas of future work that naturally follow the

results we discussed here.

Non-convexity - In Section 3, we explained that one of the requirements of our framework was the log-concavity of the V distribution. However, this is technically not required from the perspective of our formulation. Indeed, even for non-log-concave V, our framework can potentially find maps to push to V, but may ultimately require more involved optimization procedures amenable to non-convex optimization problems.

Furthermore, while some algorithms result in better approximations of the map, they may take significantly longer to compute. And while some strategies rely on less expressive approximations of the mapping model, they are more amenable to parallelization and scale better with increasing availability of hardware in terms of computation time. In general, our strategy leans more towards the latter, especially in a world that is certainly moving towards using deep neural networks as map approximations. That being said, one very interesting point of future work for this general formulation is thinking about how we might incorporate deep networks into our problem as a map parameterization. This obviously leads to the loss of convexity of our problem, among other complications, but the area of non-convex optimization theory itself is a growing field of interest that is consistently becoming theoretically richer with every passing month, especially with the recent resurgence in interest in deep learning [36].

Multiple GPUs on Multiple Hosts - A fairly straightforward improvement we can make to our CUDA implementation is to extend its capabilities to be able to accommodate distributed computing over multiple GPUs across multiple hosts. As of the writing of this document, this is still a work in progress, but it would almost certainly improve our performance for especially large problems, and make even larger problems computationally feasible. One thing to consider though, when distributing computation across GPUs, is the tradeoff between the gains achieved through finer distribution of computation, and the overhead associated with engaging more GPU devices. For problems below a certain size, it may not be beneficial to engage too many GPUs, as the overhead associated with moving data to/from that many

devices, or launching CUDA kernels on many devices, can overshadow any gains we may potentially see from the parallelization in the first place. That being said, the best way to really gauge where those tradeoffs stand is to perform an exhaustive set of benchmarks across many different system configurations and empirically determine where adding more hardware really begins to prove advantageous as a function of problem size.

Future Work in Reinforcement Learning - There is a wide range of future work that can take place in the context of reinforcement learning as described in Section 5.4. While the preliminary results in this context are not necessarily the best thus far, we feel they are certainly indicative that this approach seems promising, although it requires much more future research to really flesh out the details. Furthermore, not only does the optimal transportation framework allow for an alternative means to estimate posterior expectations of the reward model parameters, but since we are capable of quickly drawing samples from the posterior, it also allows for potential selection of trajectory queries through information-theoretic characterization of the posterior distribution.

The framework thus far relies on generating trajectories arbitrarily, without considering whether certain trajectory queries are potentially more informative than others from an information-theoretic perspective. However, it stands to reason that some queries may be more effective in providing information about the user's reward function relative to others. This can potentially be quantified by generating M candidate trajectory pairs, and then approximating the expected reduction in posterior entropy, $\mathbb{E}_Y\left[-\int_{\mathcal{X}} f(x|Y)\log f(x|Y)dx\right]$, for each trajectory pair using transport maps, and then only querying the user with the one that results in the largest drop in entropy (or alternatively, the first few "top ranked" queries in this regard). The queries that result in the largest drop in entropy would theoretically represent the "most informative" queries out of the candidates. As there are only 2 discrete values for Y, this problem would involve only computing 2 transport maps per query.

There exist several ways to use samples from a distribution to estimate the entropy [58],

but there is an interesting way this can be done that further takes advantage of our optimal transport maps. For every m^{th} candidate query we can generate a transport map to the m^{th} potential posterior termed S_m, generate N samples from each posterior X_n^m, $n = 1,\ldots,N$, $m = 1,\ldots,M$, and then create another transport map that pushes each m^{th} batch of these samples to a standard Gaussian S_m^{DE}. S_m^{DE}, $m = 1,\ldots,M$ can then be used as a density-estimation map similar to the process described in Section 5.2, which can then directly be used to estimate the posterior entropy with samples drawn from each posterior.

Optimizing the query strategy and choosing the most informative queries is of utmost importance as queries to a human can be a relatively expensive operation: queries that require active human participation can take a significant amount of time, and can induce fatigue in the user resulting in query responses that decline in quality over time. Therefore, minimizing the number of queries we need to make to acquire sufficient information about the reward function is quite an important task indeed.

Previous work has certainly attempted to make similar optimizations to the query strategy as well, such as in [15], where are ensemble of deep learning models representative of the reward function model are used to approximate a variance in trajectory rewards. Or in [24], where the construction of ideally informative query trajectories is formulated as an optimization problem in itself. But it is still a problem that eludes a concrete "best" solution overall, with different approaches resulting in a tradeoff between increasingly broad approximation and computational feasibility, so this is definitely an interesting potential topic of future work in this context.

Physical Interpretations and Connections to Deep Learning The original Monge-Kantorovich formulation of optimal transportation very much had a physical interpretation to the problem - moving a pile of earth from one shape to another. As we discussed in Section 3.4.1, and as previously described in [38], there is also a theoretical relationship between optimal transport and computational thermodynamics. Continuing to think about how transport

is inherently related to physical phenomenon is an ongoing discussion that can potentially reveal additional interesting insights into the process of computing transport maps, similar to our discussion in 3.4.1. Furthermore, the sequential nature of the compositional mapping problem discussed in this work has loose similarity to a deep neural network, as a cascading chain of non-linear functions. Indeed, works like [5] are beginning to uncover what might be a fundamental relationship between optimal transportation and modern deep learning techniques, which is quite the exciting prospect indeed. If such a relationship can be established with more solid footing, perhaps the rich theoretical history behind optimal transportation can be effectively applied to deep learning as well, to an extent.

In conclusion, this dissertation attempts to provide several valuable insights into the historically significant field of optimal transportation, and its place in the modern machine-learning-oriented world. There are certainly still many questions that must be explored when deciding where exactly optimal transport, and our formulation specifically, fits in with the myriad other machine learning algorithms we have these days, but they are questions that I truly believe can lead to some fantastic developments in our technological evolution. With publicly-available distributed computational resources becoming more and more available to the general public, it is also of critical importance to begin extending the rich theoretical underpinnings of optimal transport to distributed contexts as well, especially if we may want to apply transport to applications that may require the luxury of real-time constraints, or human-computer interfaces in general. It is certainly the hope of this dissertation author that this work acts as yet another stepping stone in the right direction in that regard.

Bibliography

[1] Riad Akrour, Marc Schoenauer, and Michele Sebag. Preference-based reinforcement learning. In *Choice Models and Preference Learning Workshop at NIPS*, volume 11, 2011.

[2] Luigi Ambrosio, Nicola Gigli, and Giuseppe Savaré. *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media, 2008.

[3] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[4] Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to mcmc for machine learning. *Machine learning*, 50(1-2):5–43, 2003.

[5] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[6] Dominique Bakry and Michel Émery. Diffusions hypercontractives. In *Séminaire de Probabilités XIX 1983/84*, pages 177–206. Springer, 1985.

[7] Rajesh Bansal and Tamer Başar. Simultaneous design of measurement and control strategies for stochastic systems with feedback. *Automatica*, 25(5):679–694, 1989.

[8] Dimitri P Bertsekas and John N Tsitsiklis. *Parallel and distributed computation: numerical methods*, volume 23. Prentice hall Englewood Cliffs, NJ, 1989.

[9] Mathieu Blondel, Vivien Seguy, and Antoine Rolet. Smooth and sparse optimal transport. *arXiv preprint arXiv:1710.06276*, 2017.

[10] Nicolas Bonnotte. From Knothe's Rearrangement to Brenier's Optimal Transport Map. *SIAM Journal on Mathematical Analysis*, 45(1):64–87, jan 2013.

[11] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[12] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[13] Yann Brenier. Décomposition polaire et réarrangement monotone des champs de vecteurs. *CR Acad. Sci. Paris Sér. I Math.*, 305:805–808, 1987.

[14] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[15] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, pages 4299–4307, 2017.

[16] Todd P. Coleman, Justin Tantiongloc, Alexis Allegra, Diego Mesa, Dae Kang, and Marcela Mendoza. Diffeomorphism learning via relative entropy constrained optimal transport. *IEEE Global Conference on Signal and Information Processing*, 2016.

[17] Nicolas Courty, Rémi Flamary, and Devis Tuia. Domain adaptation with regularized optimal transport. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 274–289. Springer, 2014.

[18] Nicolas Courty, Rémi Flamary, Devis Tuia, and Alain Rakotomamonjy. Optimal transport for domain adaptation. *arXiv preprint arXiv:1507.00504*, 2015.

[19] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[20] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems*, pages 2292–2300, 2013.

[21] Marco Cuturi and Arnaud Doucet. Fast computation of wasserstein barycenters. In *International Conference on Machine Learning*, pages 685–693, 2014.

[22] Arnaud Dessein, Nicolas Papadakis, and Jean-Luc Rouas. Regularized optimal transport and the rot mover's distance. *arXiv preprint arXiv:1610.06447*, 2016.

[23] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Openai baselines. https://github.com/openai/baselines, 2017.

[24] Anca D Dragan Dorsa Sadigh, Shankar Sastry, and Sanjit A Seshia. Active preference-based learning of reward functions. In *Robotics: Science and Systems (RSS)*, 2017.

[25] Tarek A El Moselhy and Youssef M Marzouk. Bayesian inference with optimal maps. *Journal of Computational Physics*, 231(23):7815–7850, 2012.

[26] Oriel Frigo, Neus Sabater, Vincent Demoulin, and Pierre Hellier. Optimal transportation for example-guided color transfer. In *Asian Conference on Computer Vision*, pages 655–670. Springer, 2014.

[27] Wilfrid Gangbo and Robert J McCann. The geometry of optimal transportation. *Acta Mathematica*, 177(2):113–161, 1996.

[28] Aude Genevay, Marco Cuturi, Gabriel Peyré, and Francis Bach. Stochastic optimization for large-scale optimal transport. In *Advances in Neural Information Processing Systems*, pages 3440–3448, 2016.

[29] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[30] Siva Gorantla and Todd Coleman. Information-theoretic viewpoints on optimal causal coding-decoding problems. *arXiv preprint arXiv:1102.0250*, 2011.

[31] Kristen Grauman and Trevor Darrell. Fast contour matching using approximate earth mover's distance. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2004.

[32] Steven Haker, Lei Zhu, Allen Tannenbaum, and Sigurd Angenent. Optimal mass transport for registration and warping. *International Journal of computer vision*, 60(3):225–240, 2004.

[33] Ahnaf Rashik Hassan, Syed Khairul Bashar, and Mohammed Imamul Hassan Bhuiyan. On the classification of sleep states by means of statistical and spectral features from single channel electroencephalogram. In *ICACCI*, pages 2238–2243, 2015.

[34] Yu-Chi Ho. Team decision theory and information structures. *Proceedings of the IEEE*, 68(6):644–654, 1980.

[35] Michael Horstein. Sequential transmission using noiseless feedback. *IEEE Transactions on Information Theory*, 9(3):136–143, 1963.

[36] Prateek Jain and Purushottam Kar. Non-convex optimization for machine learning. *Foundations and Trends® in Machine Learning*, 10(3-4):142–336, 2017.

[37] Bruno Jedynak, Peter I Frazier, and Raphael Sznitman. Twenty questions with noise: Bayes optimal policies for entropy loss. *Journal of Applied Probability*, 49(1):114–136, 2012.

[38] Richard Jordan, David Kinderlehrer, and Felix Otto. The variational formulation of the fokker–planck equation. *SIAM journal on mathematical analysis*, 29(1):1–17, 1998.

[39] Dae Y Kang, Pamela N DeYoung, Atul Malhotra, Robert L Owens, and Todd P Coleman. A state space and density estimation framework for sleep staging in obstructive sleep apnea. *IEEE Transactions on Biomedical Engineering*, 65(6):1201–1212, 2018.

[40] Sanggyun Kim, Rui Ma, Diego Mesa, and Todd P Coleman. Efficient bayesian inference methods via convex optimization and optimal transport. In *ISIT*, 2013.

[41] Sanggyun Kim, Diego Mesa, Rui Ma, and Todd P Coleman. Tractable fully bayesian inference via convex optimization and optimal transport theory. *arXiv preprint arXiv:1509.08582*, 2015.

[42] Sanggyun Kim, Christopher J Quinn, Negar Kiyavash, and Todd P Coleman. Dynamic and succinct statistical analysis of neuroscience data. *Proceedings of the IEEE*, 2014.

[43] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[44] Ankur A Kulkarni and Todd P Coleman. An optimizer's approach to stochastic control problems with nonclassical information structures. *IEEE Transactions on Automatic Control*, 60(4):937–949, 2015.

[45] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[46] Peihua Li, Qilong Wang, and Lei Zhang. A novel earth mover's distance methodology for image matching with gaussian mixture models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1689–1696, 2013.

[47] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[48] R. Ma and T.P. Coleman. Generalizing the posterior matching scheme to higher dimensions via optimal transportation. In *Allerton*, 2011.

[49] Youssef Marzouk, Tarek Moselhy, Matthew Parno, and Alessio Spantini. Sampling via measure transport: An introduction. *Handbook of Uncertainty Quantification*, pages 1–41, 2016.

[50] Martin McCormick, Rui Ma, and Todd P Coleman. An analytic spatial filter and a hidden markov model for enhanced information transfer rate in eeg-based brain computer interfaces. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 602–605. IEEE, 2010.

[51] Diego Mesa, Sanggyun Kim, and Todd Coleman. A scalable framework to transform samples from one continuous distribution to another. In *Information Theory (ISIT), 2015 IEEE International Symposium on*, pages 676–680. IEEE, 2015.

[52] Diego Mesa, Justin Tantiongloc, Marcela Mendoza, and Todd P. Coleman. A distributed framework for the construction of transport maps. *Neural Computation*, submitted, 2018.

[53] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[54] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[55] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[56] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pages 278–287, 1999.

[57] Adam M Oberman and Yuanlong Ruan. An efficient linear programming method for optimal transportation. *arXiv preprint arXiv:1509.03668*, 2015.

[58] Liam Paninski. Estimation of entropy and mutual information. *Neural computation*, 15(6):1191–1253, 2003.

[59] Ofir Pele and Michael Werman. Fast and robust earth mover's distances. In *ICCV*, volume 9, pages 460–467, 2009.

[60] Gert Pfurtscheller, Clemens Brunner, Alois Schlögl, and FH Lopes Da Silva. Mu rhythm (de) synchronization and eeg single-trial classification of different motor imagery tasks. *NeuroImage*, 31(1):153–159, 2006.

[61] Gert Pfurtscheller, Christa Neuper, Alois Schlogl, and Klaus Lugger. Separability of eeg signals recorded during right and left motor imagery using adaptive autoregressive parameters. *IEEE transactions on Rehabilitation Engineering*, 6(3):316–325, 1998.

[62] Lei Qin, Lei Ding, and Bin He. Motor imagery classification by means of source analysis for brain–computer interface applications. *Journal of neural engineering*, 1(3):135, 2004.

[63] Julien Rabin, Sira Ferradans, and Nicolas Papadakis. Adaptive color transfer with relaxed optimal transport. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 4852–4856. IEEE, 2014.

[64] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000.

[65] Filippo Santambrogio. Optimal transport for applied mathematicians. *Birkäuser, NY*, pages 99–102, 2015.

[66] Bernhard Schmitzer. A sparse multiscale algorithm for dense optimal transport. *Journal of Mathematical Imaging and Vision*, 56(2):238–259, 2016.

[67] Wim Schoutens. *Stochastic processes and orthogonal polynomials*, volume 146. Springer Science & Business Media, 2012.

[68] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[69] Vivien Seguy, Bharath Bhushan Damodaran, Rémi Flamary, Nicolas Courty, Antoine Rolet, and Mathieu Blondel. Large-scale optimal transport and mapping estimation. *arXiv preprint arXiv:1711.02283*, 2017.

[70] Ofer Shayevitz and Meir Feder. Optimal feedback communication via posterior matching. *IEEE Transactions on Information Theory*, 57(3):1186–1222, 2011.

[71] Justin Solomon, Fernando De Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas. Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics (TOG)*, 34(4):66, 2015.

[72] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 1998.

[73] Justin Tantiongloc, Diego Mesa, Rui Ma, Sanggyun Kim, Cristian H Alzate, Jaime J Camacho, Vidya Manian, and Todd P. Coleman. An information and control framework for optimizing user-compliant human computer interfaces. *Proceedings of the IEEE*, 102(2):273–285, 2017.

[74] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.

[75] Theodoros Tsiligkaridis, Brian M Sadler, and Alfred O Hero. Collaborative 20 questions for target localization. *IEEE Transactions on Information Theory*, 60(4):2233–2252, 2014.

[76] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer, 2008.

[77] Jean Walrand and Pravin Varaiya. Optimal causal coding-decoding problems. *IEEE Transactions on Information Theory*, 29(6):814–820, 1983.

[78] Wei Wang, Dejan Slepčev, Saurav Basu, John A Ozolek, and Gustavo K Rohde. A linear optimal transportation framework for quantifying and visualizing variations in sets of images. *International journal of computer vision*, 101(2):254–269, 2013.

[79] Christian Wirth, Riad Akrour, Gerhard Neumann, and Johannes Fürnkranz. A survey of preference-based reinforcement learning methods. *The Journal of Machine Learning Research*, 18(1):4945–4990, 2017.

[80] Christian Wirth, Johannes Furnkranz, and Gerhard Neumann. Model-free preference-based reinforcement learning. In *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, pages 2222–2228, 2016.

[81] Dongbin Xiu and George Em Karniadakis. The wiener–askey polynomial chaos for stochastic differential equations. *SIAM journal on scientific computing*, 24(2):619–644, 2002.

[82] Guohun Zhu, Yan Li, and Peng Paul Wen. Analysis and classification of sleep stages based on difference visibility graphs from a single-channel eeg signal. *REM*, 806:803, 2014.